Qualys Container Security Assessment and Response

Lab Tutorial Supplement

Table of Contents

Install Container Sensor on a Docker Host

Qualys Container Sensor and the Qualys Container Security application provide continuous protection for docker applications in the DevOps pipeline, both in public cloud and on-premise environments. This includes: discovery, tracking, and vulnerability management.

System Support

Qualys Container Security supports these systems running Docker version 1.12 and above:

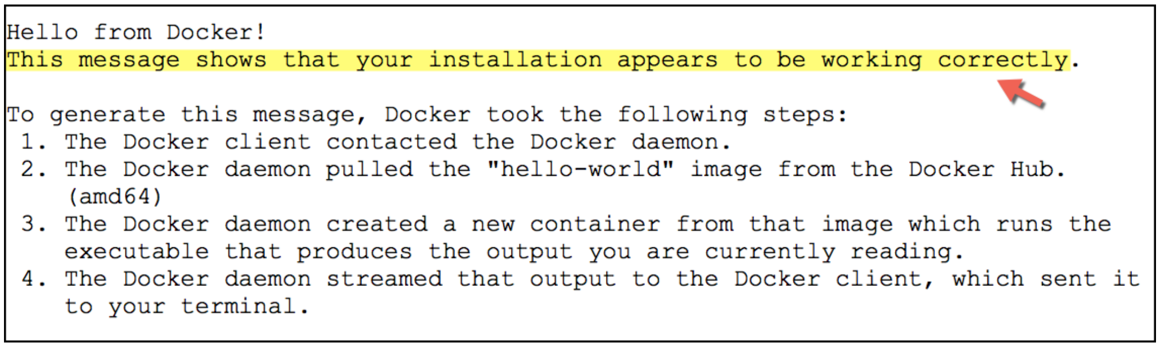
- Ubuntu
- Debian
- Fedora
- Red Hat Enterprise
- CentOS
- Mac OS
- CoreOS

You can deploy the Qualys Container Sensor on a host that has a Container Engine such as Docker installed on one of the supported operating systems listed above.

Please consult https://docs.docker.com/install/overview/ *for information on installing Docker.*

To test your docker installation, run the “hello-world” app as follows:

$ docker run hello-world



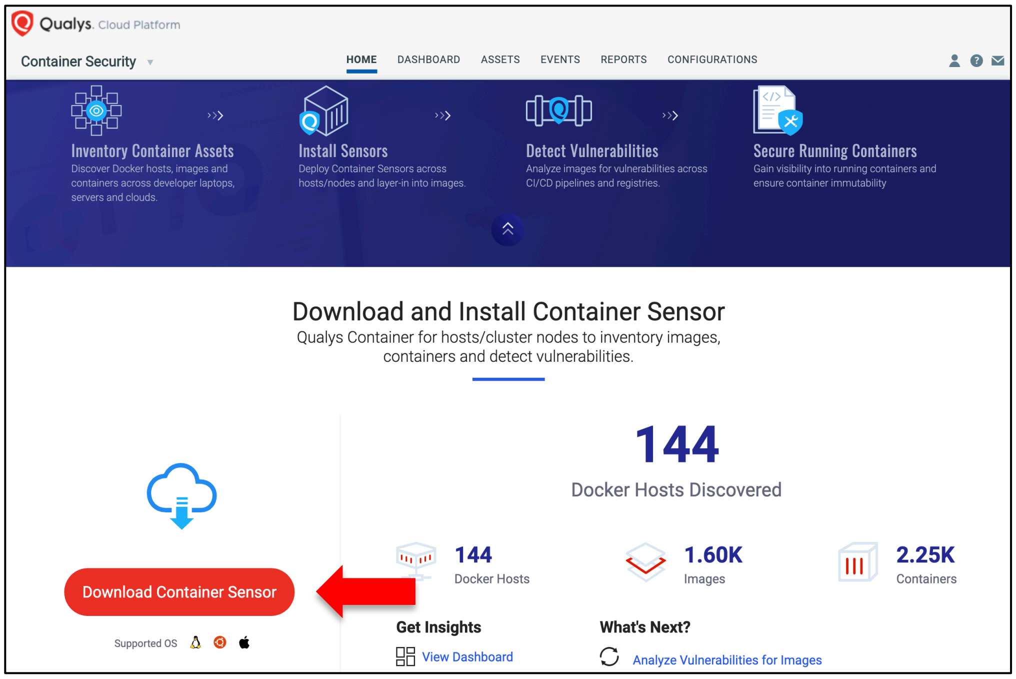
```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

You should receive the above message if your Docker installation is working fine.

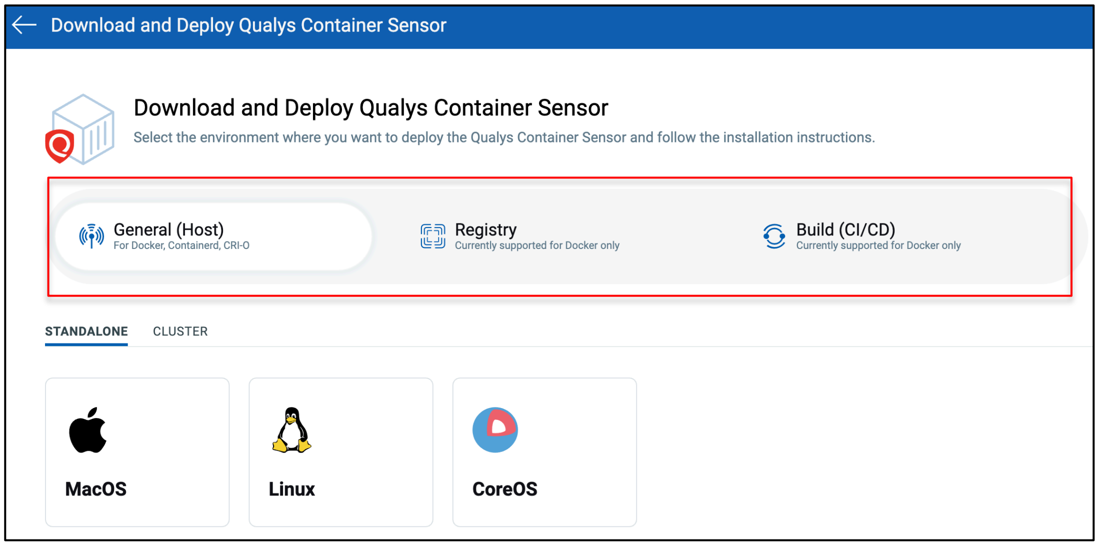
Container Sensor Installation

The Container Sensor from Qualys is packaged and delivered as a Docker Image. Simply download the Container Sensor image and deploy it as a container alongside other application containers on a Docker host.

You can download the Container Sensor image from the Container Security application “Home” page:


Qualys. Cloud Platform
Container Security
HOME
DASHBOARD
ASSETS
EVENTS
REPORTS
CONFIGURATIONS
Inventory Container Assets
Discover Docker hosts, images and containers across developer laptops, servers and clouds.
Install Sensors
Deploy Container Sensors across hosts/nodes and layer-in into images.
Detect Vulnerabilities
Analyze images for vulnerabilities across CI/CD pipelines and registries.
Secure Running Containers
Gain visibility into running containers and ensure container immutability
Download and Install Container Sensor
Qualys Container for hosts/cluster nodes to inventory images, containers and detect vulnerabilities.
144
Docker Hosts Discovered
144
Docker Hosts
1.60K
Images
2.25K
Containers
Download Container Sensor
Supported OS
Get Insights
View Dashboard
What's Next?
Analyze Vulnerabilities for Images

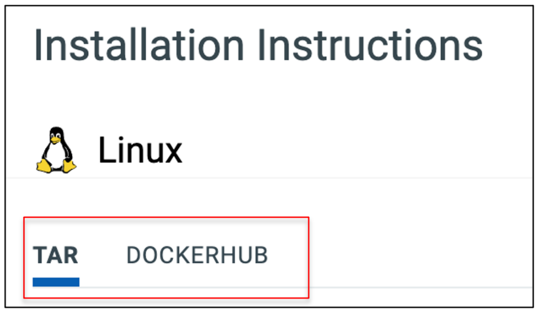
Multiple Sensor options are available for different scanning environments:


Download and Deploy Qualys Container Sensor
Download and Deploy Qualys Container Sensor
Select the environment where you want to deploy the Qualys Container Sensor and follow the installation instructions.
General (Host)
For Docker, Containerd, CRI-O
Registry
Currently supported for Docker only
Build (CI/CD)
Currently supported for Docker only
STANDALONE
CLUSTER
MacOS
Linux
CoreOS

General Sensor – scan any image outside of registry or CI/CD build.
Registry Sensor – scan images in a public or private registry.
Build (CI/CD) Sensor – scan images within your CI/CD pipeline.

***Please note the General Sensor supports Docker, Containerd and CRI-O container runtime environments. The Registry Sensor and Build (CI/CD) Sensor currently support only the Docker container runtime environment ***

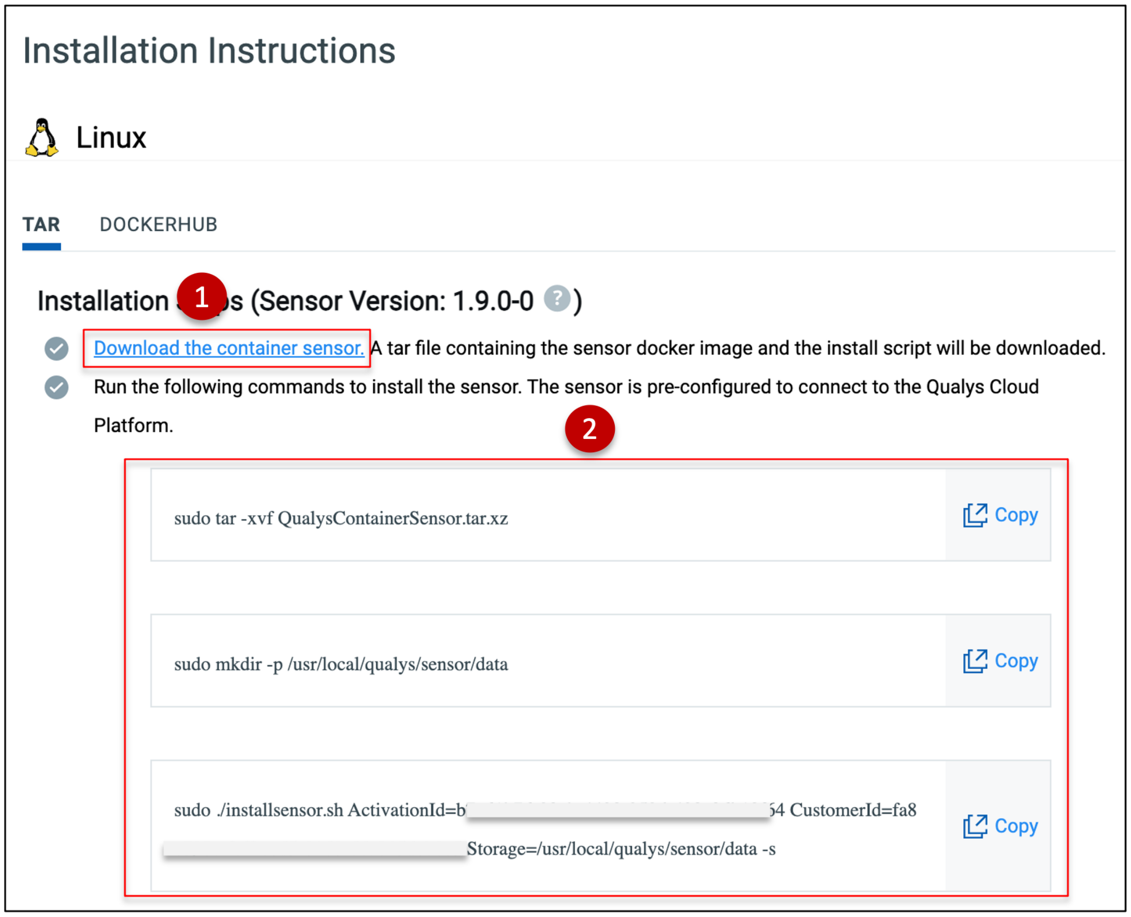
You can find more information about the sensor options in the Qualys Container Security User Guide: https://www.qualys.com/docs/qualys-container-security-user-guide.pdf



The Container Security Sensor can be installed in either of the following ways:

- Download the sensor tar file from Qualys Cloud Platform and then install it on the host
- Install the sensor from Docker Hub

For Tar, you'll need to first download the tar file to your container host (or download and then transfer the file to your container host) and then run the install commands shown on the screen, on the container host.



Installation Instructions

Linux

Installation s (Sensor Version: 1.9.0-0 ?)

- Download the container sensor. A tar file containing the sensor docker image and the install script will be downloaded.
- Run the following commands to install the sensor. The sensor is pre-configured to connect to the Qualys Cloud Platform.

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

```
sudo mkdir -p /usr/local/qualys/sensor/data
```

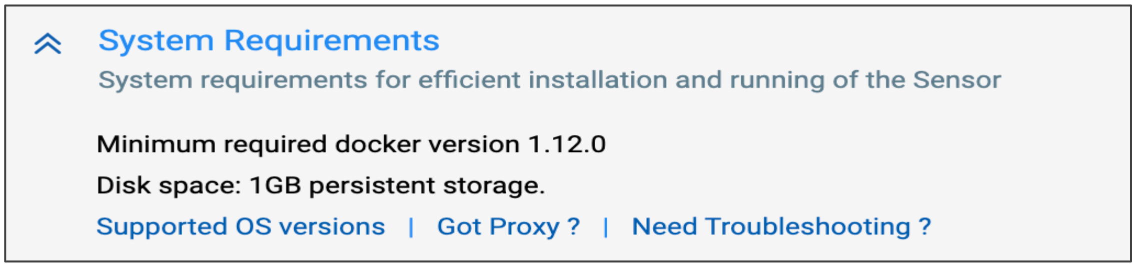
```
sudo ./installsensor.sh ActivationId=b[redacted]54 CustomerId=fa8
[redacted]Storage=/usr/local/qualys/sensor/data -s
```

– *The first command unpacks the contents of the tarball file.*

– *The second command creates the target installation directory path.*

– *The third command executes the installation script (installsensor.sh) which installs the sensor with your account's Activation Key and Customer ID.*

It's this third command that distinguishes the different types of sensor options (i.e., General, Registry, and CI/CD Sensor). This command will contain an additional parameter for the Registry Sensor (-r), and the Build or CI/CD Sensor (-c).



System Requirements

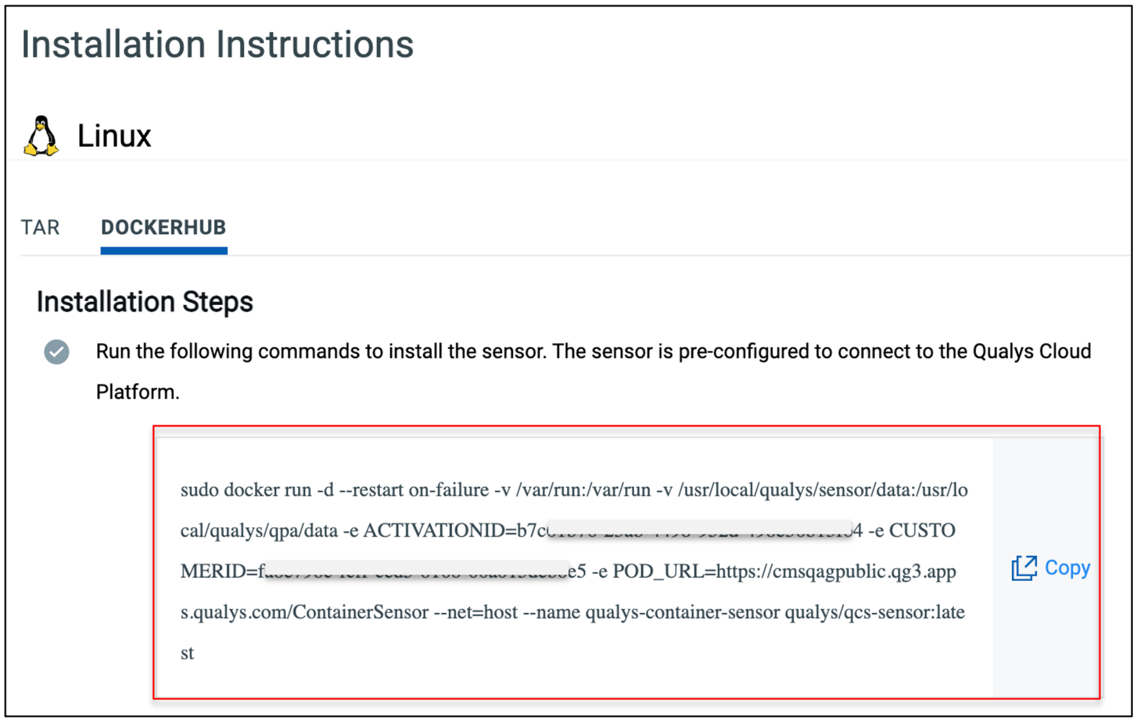
System requirements for efficient installation and running of the Sensor

Minimum required docker version 1.12.0

Disk space: 1GB persistent storage.

Supported OS versions | Got Proxy ? | Need Troubleshooting ?

Notice the “System Requirements” for installing the Sensor. Ensure that your container host meets these system requirements for a successful container sensor deployment.


Installation Instructions

Linux

Installation Steps

Run the following commands to install the sensor. The sensor is pre-configured to connect to the Qualys Cloud Platform.

```
sudo docker run -d --restart on-failure -v /var/run:/var/run -v /usr/local/qualys/sensor/data:/usr/lo
cal/qualys/qpa/data -e ACTIVATIONID=b7c[redacted]4 -e CUSTO
MERID=f[redacted]e5 -e POD_URL=https://cmsqagpublic.qg3.app
s.qualys.com/ContainerSensor --net=host --name qualys-container-sensor qualys/qcs-sensor:late
st
```

For DockerHub, you’ll run the docker run command as shown on the screen, on your container host. The installation command contains your Activation ID, Customer ID and POD_URL of your Qualys subscription.

Additional environment variables/volumes can be provided.

The Container Security Sensor on Docker Hub is available as:
qualys/qcs-sensor: <tag>
qualys/qcs-sensor:latest

You can specify the most recent tag (latest tag) in Docker Hub.

Please consult the Container Sensor Deployment Guide for more information:
https://www.qualys.com/docs/qualys-container-sensor-deployment-guide.pdf

Please note that the Docker Hub sensor image is not supported for customers on the Qualys Private Cloud Platform (PCP).

If the Container Sensor installation fails or if the Sensor keeps crashing after installation,

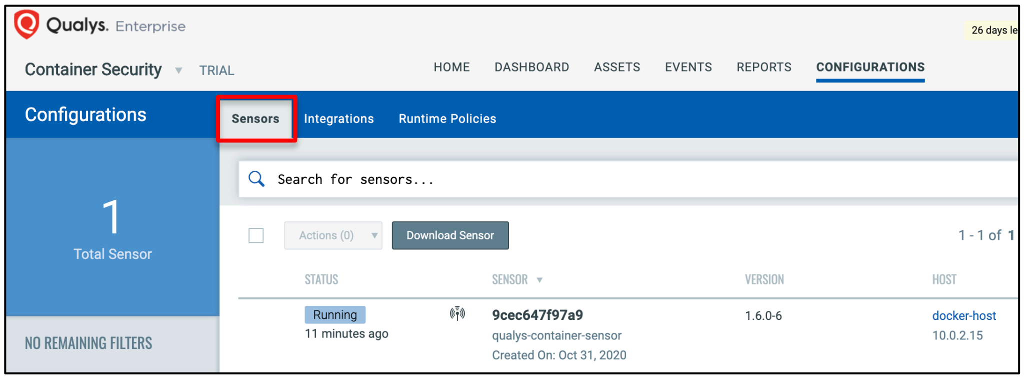
you should first consult the Sensor logs stored in the following location on the container host:
`/usr/local/qualys/sensor/data/logs/qpa.log`

*** Note – Mac OS and CoreOS use different installation commands. ***

You'll find more detailed information for Mac OS and CoreOS installations in the Container Sensor Deployment Guide:
https://www.qualys.com/docs/qualys-container-sensor-deployment-guide.pdf

Sensors Tab

After successfully installing the Container Sensor in your environment, you can see the Sensor in the "Configurations" section under the "Sensors" Tab in the CS application.



STATUS	SENSOR	VERSION	HOST
Running 11 minutes ago	9cec647f97a9 qualys-container-sensor Created On: Oct 31, 2020	1.6.0-6	docker-host 10.0.2.15

It may take a few minutes for your newly installed sensor to appear. The sensor should be in the "Running" state following a successful installation.

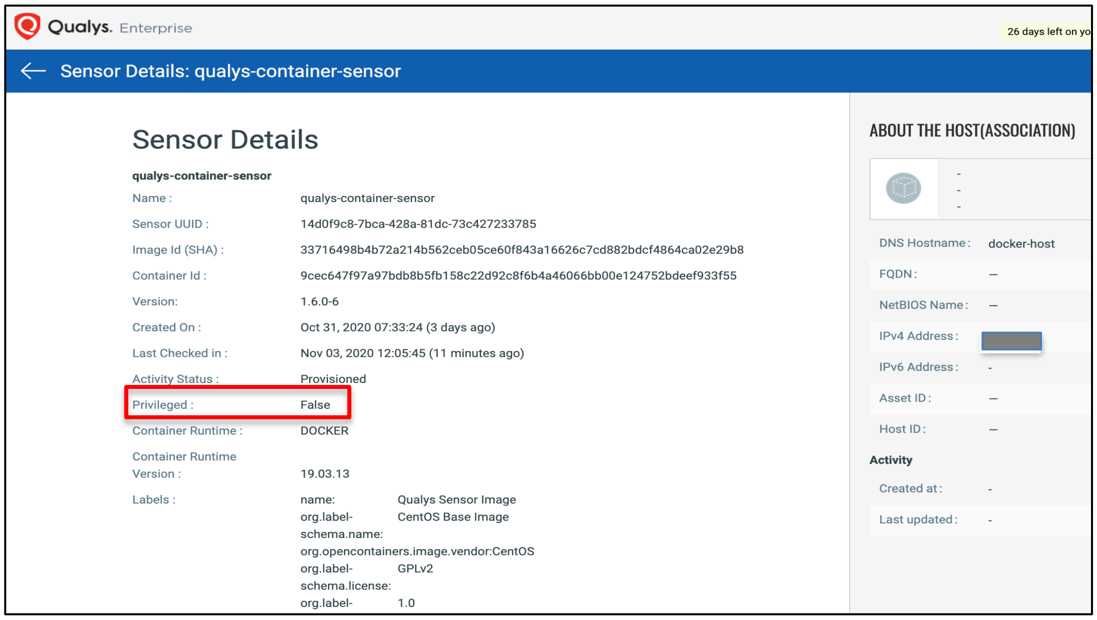
The General Sensor automatically scans its docker host for images and containers that are present and sends the collected data and metadata to the Qualys Cloud Platform for processing and assessment. By default, Container Sensor is pre-configured to communicate with the Qualys Cloud platform on TCP port 443.

Alternatively, it can be configured to support proxy servers.

You will find more information and details about proxy configuration in the Qualys Container Security User Guide:

https://www.qualys.com/docs/qualys-container-security-user-guide.pdf

You can use the "Quick Actions" menu to view Sensor details.



Sensor Details

qualys-container-sensor

Name :	qualys-container-sensor
Sensor UUID :	14d0f9c8-7bca-428a-81dc-73c427233785
Image Id (SHA) :	33716498b4b72a214b562ceb05ce60f843a16626c7cd882bdcf4864ca02e29b8
Container Id :	9cec647f97a97bdb8b5fb158c22d92c8f6b4a46066bb00e124752bdeef933f55
Version:	1.6.0-6
Created On :	Oct 31, 2020 07:33:24 (3 days ago)
Last Checked in :	Nov 03, 2020 12:05:45 (11 minutes ago)
Activity Status :	Provisioned
Privileged :	False
Container Runtime :	DOCKER
Container Runtime Version :	19.03.13
Labels :	name: Qualys Sensor Image org.label-schema.name: CentOS Base Image org.opencontainers.image.vendor:CentOS org.label-schema.license: GPLv2 org.label- 1.0

ABOUT THE HOST(ASSOCIATION)

DNS Hostname :	docker-host
FQDN :	—
NetBIOS Name :	—
IPv4 Address :	[redacted]
IPv6 Address :	-
Asset ID :	—
Host ID :	—

Activity

Created at :	-
Last updated :	-

On the Sensor Details page, you can see identification information for the sensor. Notice that the “Privileged” parameter is set to “False” which indicates that the Container sensor is installed in a non-privileged mode.

Navigate to the following URL to view the “Install Container Sensor on a Docker Host” tutorial:

PLAY http://ior.ad/7hbj

Install Container Sensor in a Cluster

To successfully install the Qualys Container Sensor in a cluster environment such as Kubernetes or Docker Swarm, the following configuration steps are required:

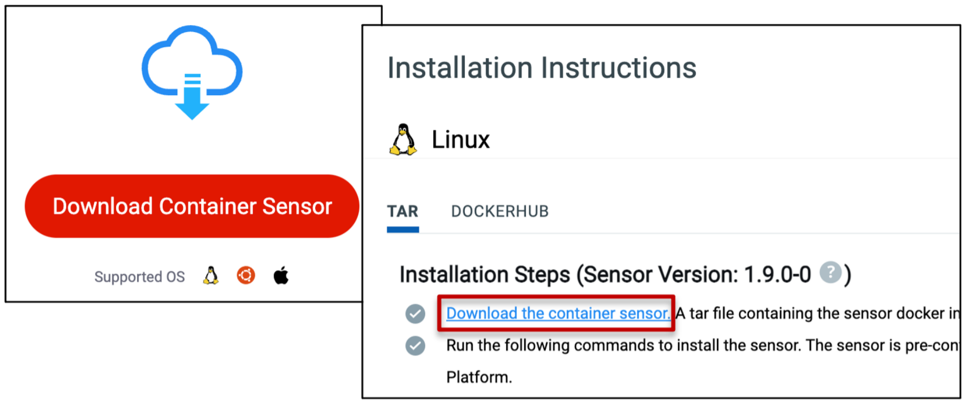
1. Download the Container Sensor image (available in the Container Security application and in Docker Hub)
2. Push the Sensor image to your public\private Registry
3. Configure the Sensor deployment template (Eg. cssensor-ds.yml for Kubernetes)
4. Deploy the Sensor (Eg. create a Daemonset for the Sensor on Kubernetes)

Sensor Deployment Requirements

- The cluster setup should be up and running
- All nodes that will be running the Sensor pod must have access to the private or docker hub registry where sensor image is stored.
- If you are using an https proxy, ensure that all cluster nodes have a valid certificate file for the sensor to communicate with the Container Management Server
- The Container Sensor must have read and write access to the persistent storage and the docker daemon socket on each cluster node

Download the Sensor Image

First, you need to download the container sensor image. This can be done from under the Container Security application.



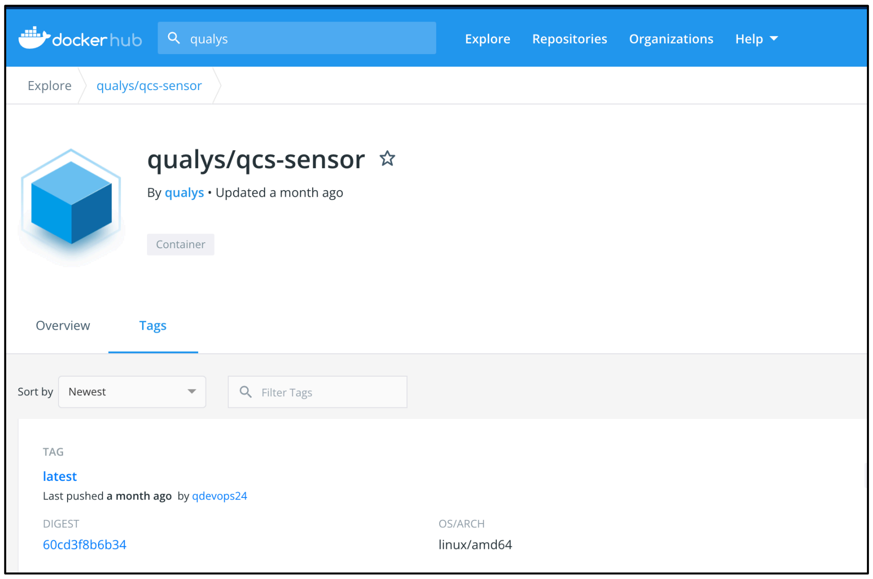
Installation Steps (Sensor Version: 1.9.0-0)

Download the container sensor. A tar file containing the sensor docker i

Run the following commands to install the sensor. The sensor is pre-con

Platform.

You can also use the Qualys Container Sensor image from Docker Hub.



qualys/qcs-sensor

By qualys • Updated a month ago

TAG: latest

Last pushed a month ago by qdevops24

DIGEST: 60cd3f8b6b34

OS/ARCH: linux/amd64

The Container Security Sensor on Docker Hub is available as:
qualys/qcs-sensor: <tag>
qualys/qcs-sensor:latest

The Docker Hub Qualys Container Sensor image can either be first downloaded and pushed to your public/private registry or referenced directly in the deployment template.

Extract and Push the Sensor Image to a Registry

If you downloaded the Sensor Image tar file, you need to extract the sensor image from the tar file using the following command:

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

Next you need to load the Sensor image and tag it.
For example:

```
sudo docker load -i qualys-sensor.tar

sudo docker tag c3fa63a818df mycloudregistry.com/container-sensor:qualys-sensor-xxx
```

Where *c3fa63a818df* is the image ID of the Sensor image and *mycloudregistry.com/container-sensor:qualys-sensor-xxx* is the target registry/repository path.

You then need to push the tagged image to a public or private Registry that is accessible on all cluster nodes.

For example:
```
sudo docker push mycloudregistry.com/container-
sensor:qualys-sensor-xxx
```

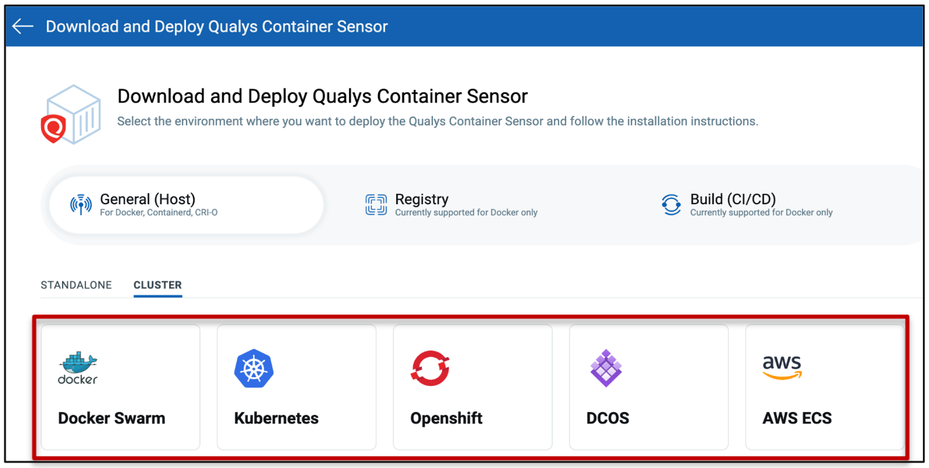
Above steps are not required if deploying directly through the Qualys Docker Hub repository.

Configure the Deployment Template

Next, you need to download and configure a deployment template for the sensor installation in the cluster.

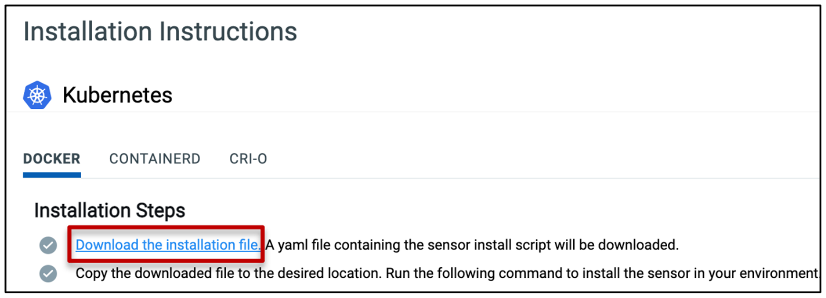
You can download deployment templates for various Container Orchestration Platforms and Container Runtime environments from the Container Security application.

Simply select the Sensor type (General, Registry or Build) and the cluster environment (Docker Swarm, Kubernetes, etc.) to download the deployment template. These templates are also included in the Sensor image tar file.


Download and Deploy Qualys Container Sensor
General (Host)
For Docker, Containerd, CRI-O
Registry
Currently supported for Docker only
Build (CI/CD)
Currently supported for Docker only
STANDALONE
CLUSTER
Docker Swarm
Kubernetes
Openshift
DCOS
AWS ECS

Download and Deploy Qualys Container Sensor

Select the environment where you want to deploy the Qualys Container Sensor and follow the installation instructions.

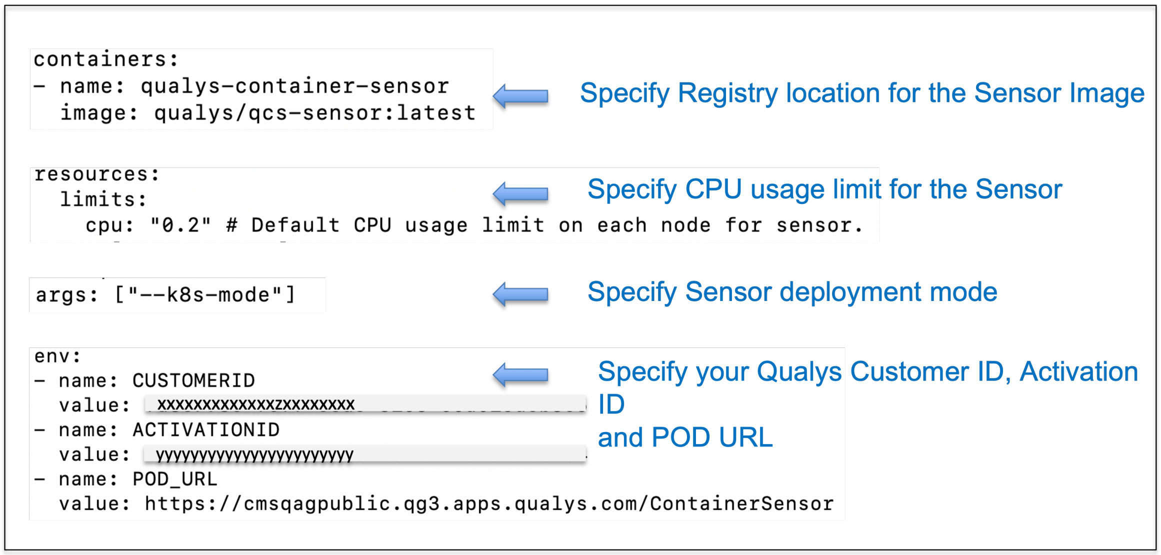


Installation Steps

- Download the installation file. A yaml file containing the sensor install script will be downloaded.
- Copy the downloaded file to the desired location. Run the following command to install the sensor in your environment

In addition, the deployment templates can also be downloaded from a public GitHub repository maintained by Qualys https://github.com/Qualys/cs_sensor

For Kubernetes, you need to use the **cssensor-ds.yml** file.



```
containers:
- name: qualys-container-sensor
  image: qualys/qcs-sensor:latest

resources:
  limits:
    cpu: "0.2" # Default CPU usage limit on each node for sensor.

args: ["--k8s-mode"]

env:
- name: CUSTOMERID
  value: xxxxxxxxxxxxxxzxxxxxxxxx
- name: ACTIVATIONID
  value: yyyyyyyyyyyyyyyyyyyyyyy
- name: POD_URL
  value: https://cmsqagpublic.qg3.apps.qualys.com/ContainerSensor
```

Specify Registry location for the Sensor Image
Specify CPU usage limit for the Sensor
Specify Sensor deployment mode
Specify your Qualys Customer ID, Activation ID
and POD URL

You need to include necessary information such as the registry and the repository path for the sensor image, the Activation ID and Customer ID of your Qualys account, the deployment mode (General/CICD/Registry) and other information specific to your environment and deployment type, in the template.

You need to ensure that formatting provided in the template is maintained and you do not remove/comment out the required sections and arguments in the template, for the deployment to work correctly.

Please consult the Container Sensor Deployment Guide for more information

https://www.qualys.com/docs/qualys-container-sensor-deployment-guide.pdf

Create a Daemonset

The Qualys Container Sensor is deployed as a daemonset just as any other application container in the Kubernetes environment.

The Sensor deployment is initiated from the master node. Once you have modified the cssensor-ds.yml file, run the following command on Kubernetes master to create a DaemonSet:

```
kubectl create -f cssensor-ds.yml
```

Kubernetes automatically maintains the specified number of Container Sensor instances in the cluster.

Navigate to the following URL to view the "Container Sensor Installation in Kubernetes" tutorial:

http://ior.ad/7bwh

Visibility into Container Projects

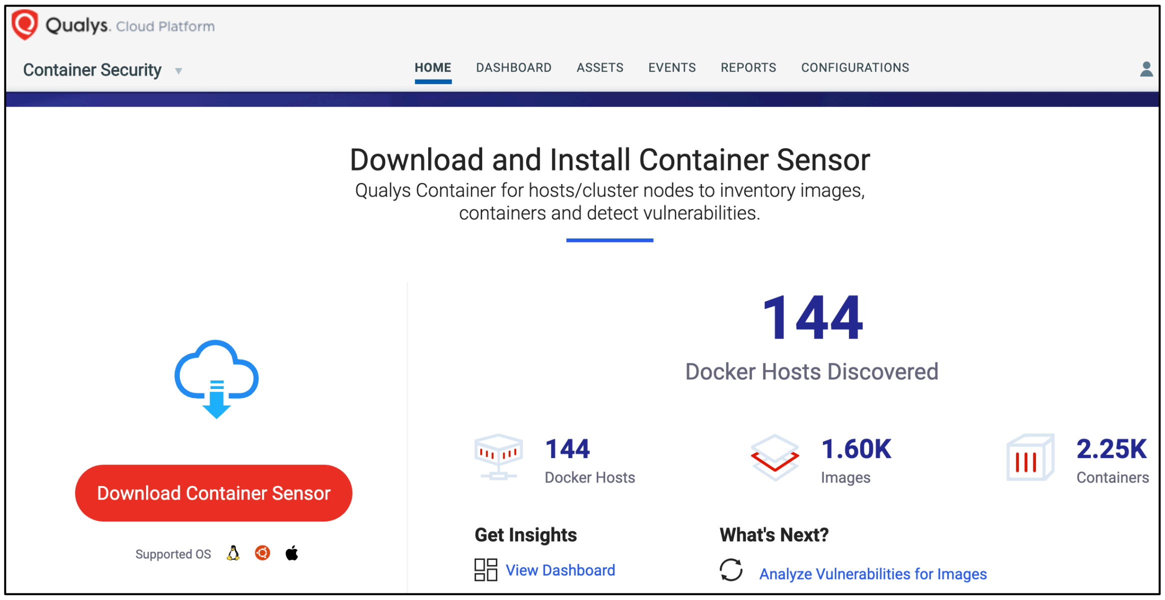
The first use case for Qualys Container Security is about discovery, inventory, and near-real time tracking of container environments. If you are doing Vulnerability Management using either Scanner appliances or Cloud Agents, we can tell where you have your containers running and what they are as well.

We have multiple Information Gathering type of QIDs in our knowledgebase for identifying this information.

QID 45367	Docker Containers Status Enumerated
QID 45434	containerd Version detected
QID 370440	Docker Running Container Enumerated
QID 48030	Qualys Container Security Sensor Detected

All above QIDs need **authenticated scan for discovery if scanning using a scanner appliance.*

A summary of all discovered Docker hosts and their image and container inventory is readily available on the Container Security application "Home" page.



Download and Install Container Sensor

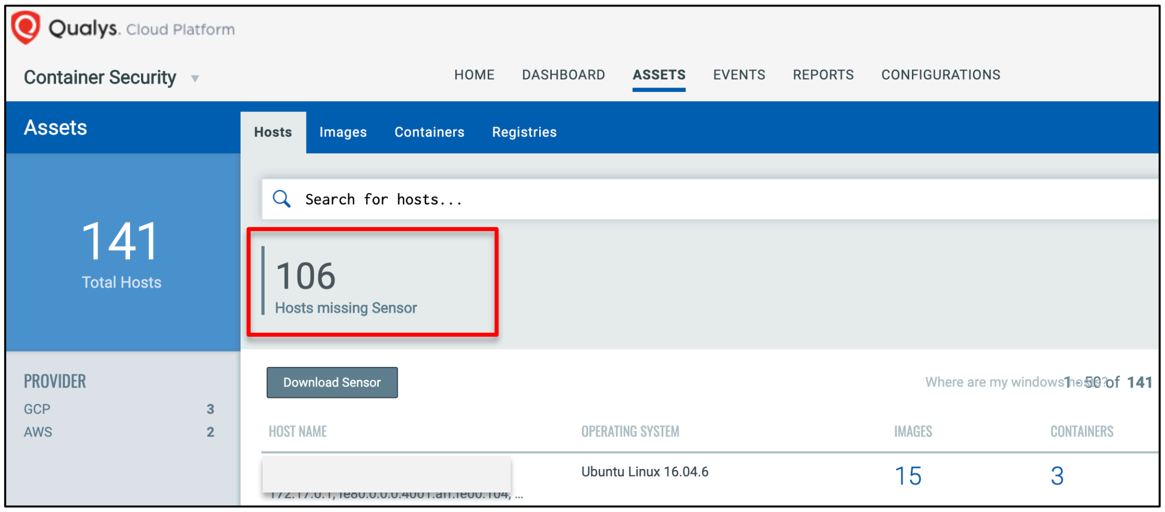
Qualys Container for hosts/cluster nodes to inventory images, containers and detect vulnerabilities.

144
Docker Hosts Discovered

144 Docker Hosts
1.60K Images
2.25K Containers

Hosts Missing Sensor

The "Hosts" Tab under "Assets" section in the Container Security application lists all discovered Docker hosts in your environment.

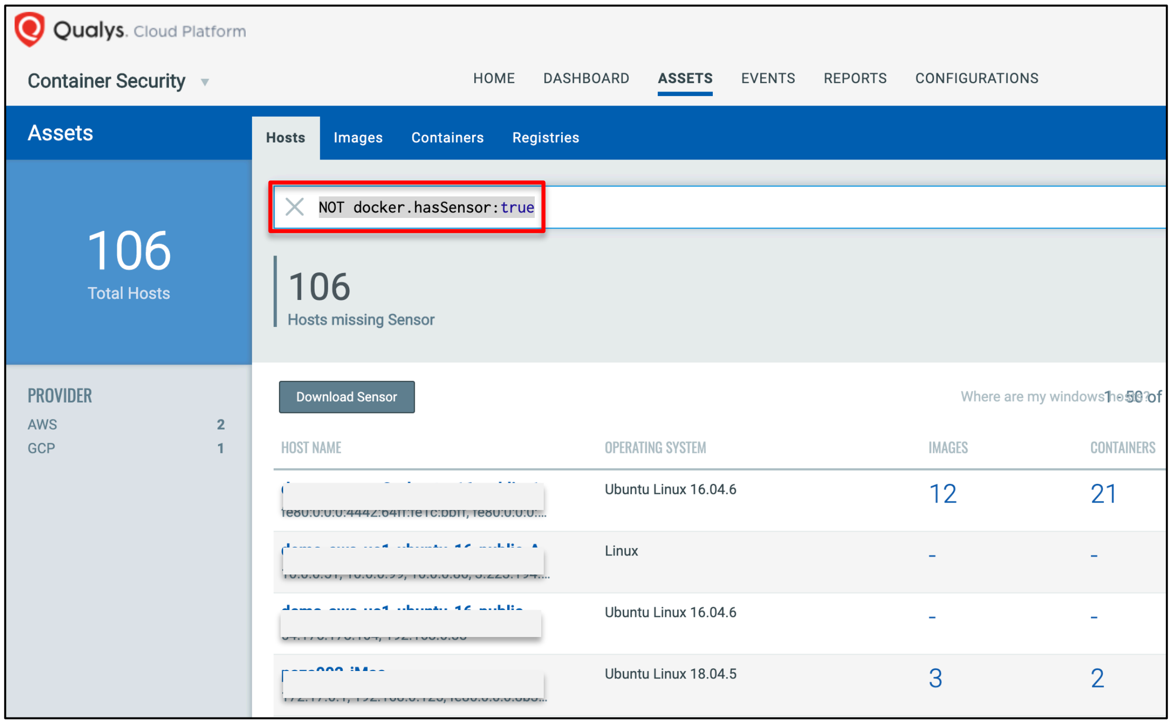


PROVIDER

GCP	3
AWS	2

HOST NAME	OPERATING SYSTEM	IMAGES	CONTAINERS
[redacted] 172.17.0.1, fe80:0:0:0:4001:aff:fe80:104, ...	Ubuntu Linux 16.04.6	15	3

Clicking on the “Hosts missing Sensor” option will automatically create a search query to filter all Docker hosts missing the Container Sensor



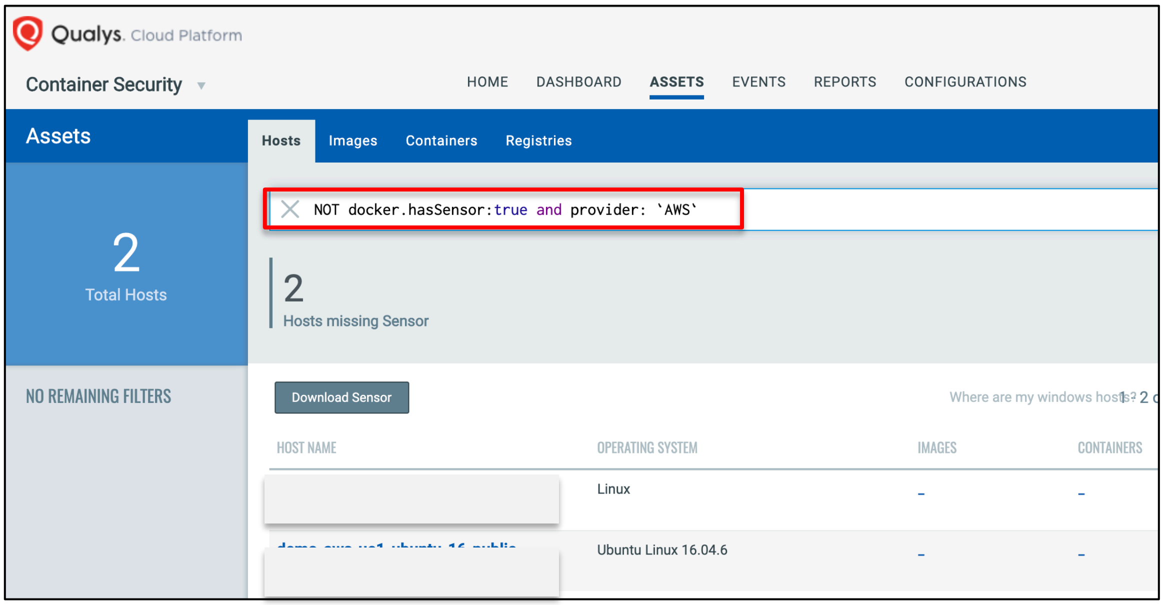
```
NOT docker.hasSensor:true
```

PROVIDER

AWS	2
GCP	1

HOST NAME	OPERATING SYSTEM	IMAGES	CONTAINERS
[redacted] fe80:0:0:0:4442:64ff:fe1c:bbff, fe80:0:0:0:...	Ubuntu Linux 16.04.6	12	21
[redacted] 10.0.0.51, 10.0.0.99, 10.0.0.80, 3.223.194...	Linux	-	-
[redacted] 54.175.175.104, 192.168.0.38	Ubuntu Linux 16.04.6	-	-
[redacted] 172.17.0.1, 192.168.0.120, fe80:0:0:0:0b0...	Ubuntu Linux 18.04.5	3	2

Further, if you have deployed the Qualys Cloud Connector for AWS, Azure or Google in the Global Asset Inventory application, we can automatically identify if any of your Docker hosts are hosted in a public cloud infrastructure such as AWS, Azure or GCP and if they are missing Container Security.



```
NOT docker.hasSensor:true and provider: `AWS`
```

HOST NAME	OPERATING SYSTEM	IMAGES	CONTAINERS
	Linux	-	-
	Ubuntu Linux 16.04.6	-	-

Once you have this visibility, you can deploy the Container Sensor on any hosts missing Container Security and thus eliminate blind spots in your environment.

Track Image Location

This visibility also extends to tracking the image source and identifying unknown Registry locations. the Container Security application will have the necessary metadata as you scan your images in the CI pipeline in the build phase. This information can be used to identify images that made it through the CI phase but are not found in any of the scanned Registries or in the CD pipeline.

This may be an indication that such images may be stored in an unknown image repository or a Registry location that the security team is unaware of. So, identifying images in such unprotected Registry locations becomes important. You can use search queries in the Container Security application to quickly identify such information.

The below query lists images that were scanned by the Build (CI/CD) Sensor but are not present in any of the Registry locations scanned by the Registry Sensor or located on any host scanned by the General sensor:

```
source:CICD and not (source:REGISTRY and source:GENERAL )
```

Track the Image Source

We can also verify which images were scanned in the secure build pipeline and which ones were not. Images that were not scanned in the build phase but are present in a Registry scanned by the Registry Sensor or deployed in a production environment and scanned by the General Sensor, can become a concern. It could be an indication that someone introduced a rogue image from outside of the pipeline into the production environment. So, identifying such images gives you the visibility you need to ensure that people are following the right guidelines that are laid out and only compliant images get pushed to the registry and into the production environment.

The below query can be useful to quickly identify images that were not scanned in the CICD pipeline but are present in a scanned Registry or identified by the General sensor:

```
not source:CICD and (source:REGISTRY and source:GENERAL )
```

You can use such queries to create dashboard widgets in the Container Security application to display and refresh this information automatically.

Navigate to the following URL to view the "Visibility into Container Projects" tutorial:

PLAY http://ior.ad/7hct

Assess Containerized Applications

After installing the General Sensor, it automatically scans images and containers present on the container host and sends the metadata to the Qualys Cloud Platform for processing and assessment. This lab will use a simple containerized application to highlight some of the functions and features of the Container Security application.

Bodgeit Store Application

The Bodgeit Store Web app is used as a target in the Qualys Web Application Scanning training class. By building the Bodgeit app on the same docker host, where you installed the Qualys Container Sensor, you will see how the sensor captures inventory and events from application images and containers.

Application Setup

Container applications are created from a **Docker File** that specifies all the application layers required to build an Image. The bodgeit application used in this lab is also built using a Docker file as shown below

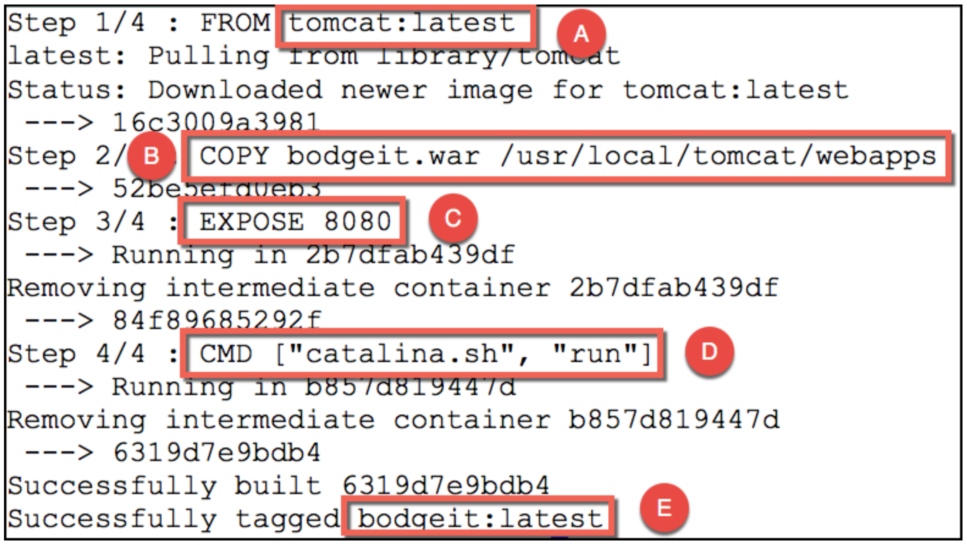
```
# Use an official Tomcat runtime as a parent image
FROM tomcat:latest

# Copy "bodgeit.war" file to the tomcat webapps directory.
COPY bodgeit.war /usr/local/tomcat/webapps

# Make port 8080 available to the world outside this container.
EXPOSE 8080

# Run tomcat when the container launches.
CMD ["catalina.sh", "run"]
```

Build Bodgeit Image

The docker “build” command creates an image from the layers identified in the above Docker file.

```
$ docker build -t bodgeit .
```

This command need to be run from the directory containing the files required by the container application (bodgeit.war file in this instance). The dot or period at the end, specifies the present working directory (pwd) as the source of the docker build.
The -t parameter assigns the “bodgeit” tag to the image that is created, making it easier to identify.



```
Step 1/4 : FROM tomcat:latest
latest: Pulling from library/tomcat
Status: Downloaded newer image for tomcat:latest
 ---> 16c3009a3981
Step 2/4 : COPY bodgeit.war /usr/local/tomcat/webapps
 ---> 52be5efd0eb3
Step 3/4 : EXPOSE 8080
 ---> Running in 2b7dfab439df
Removing intermediate container 2b7dfab439df
 ---> 84f89685292f
Step 4/4 : CMD ["catalina.sh", "run"]
 ---> Running in b857d819447d
Removing intermediate container b857d819447d
 ---> 6319d7e9bdb4
Successfully built 6319d7e9bdb4
Successfully tagged bodgeit:latest
```

A successful build will reflect the results displayed above:

A. The latest version of tomcat is downloaded.

B. The bodgeit.war file has been copied to the tomcat webapps directory.

C. The application is exposed on port 8080.

D. Run tomcat application (catalina.sh).

E. Application successfully tagged as bodgeit:latest

Run bodgeit Application

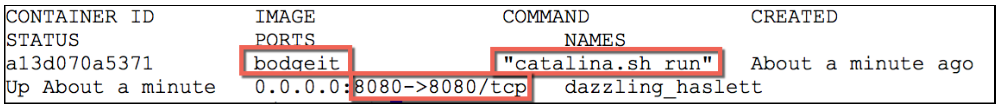
The docker "run" command indicated below will create a container instance from the bodgeit image :

```
$ docker run -d -p 8080:8080 bodgeit
```

The -d parameter will run the app in "detached" mode, allowing it to operate in the background.
The -p parameter connects port 8080 of the docker container to port 8080 of the docker host. Other users or applications will use port 8080 to access Bodgeit Store application services.

To following command indicates if your container application is running:

```
$ docker ps
```



```
CONTAINER ID        IMAGE                    COMMAND             CREATED
STATUS              PORTS                         NAMES
a13d070a5371        bodgeit                  "catalina.sh run"   About a minute ago
Up About a minute   0.0.0.0:8080->8080/tcp        dazzling_haslett
```

The output should resemble the illustration displayed above.

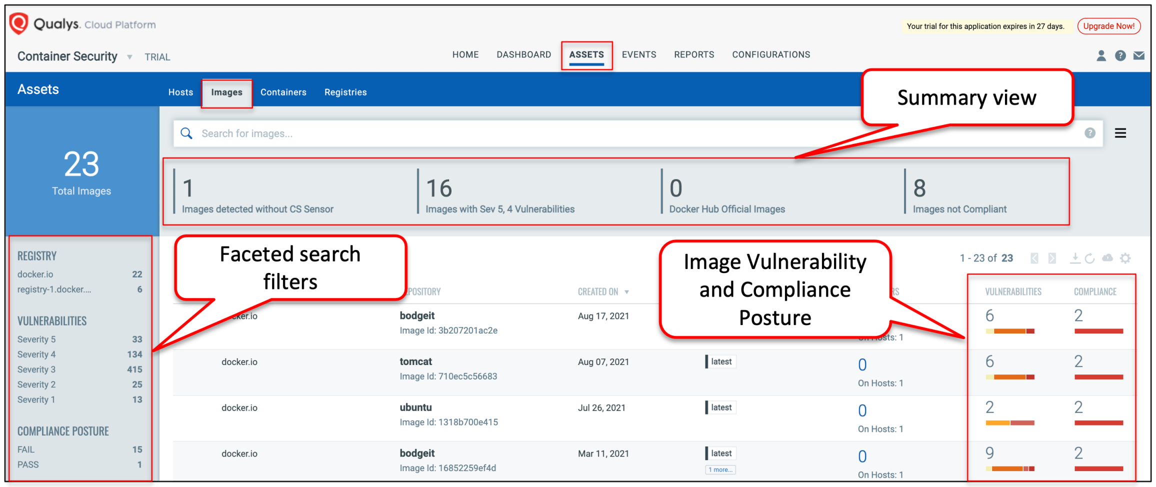
Analyse Results

The Qualys Container Sensor captures the bodgeit application events and information and sends them to the Qualys Cloud Platform, where you can analyze and act upon the results and findings.

Assets

The "Assets" section lists the Images and Containers discovered along with their metadata information like ports, networks, services, users, installed software, etc. Images are listed along with associated vulnerabilities and containers.

The "Search" field can be used to locate a specific image.



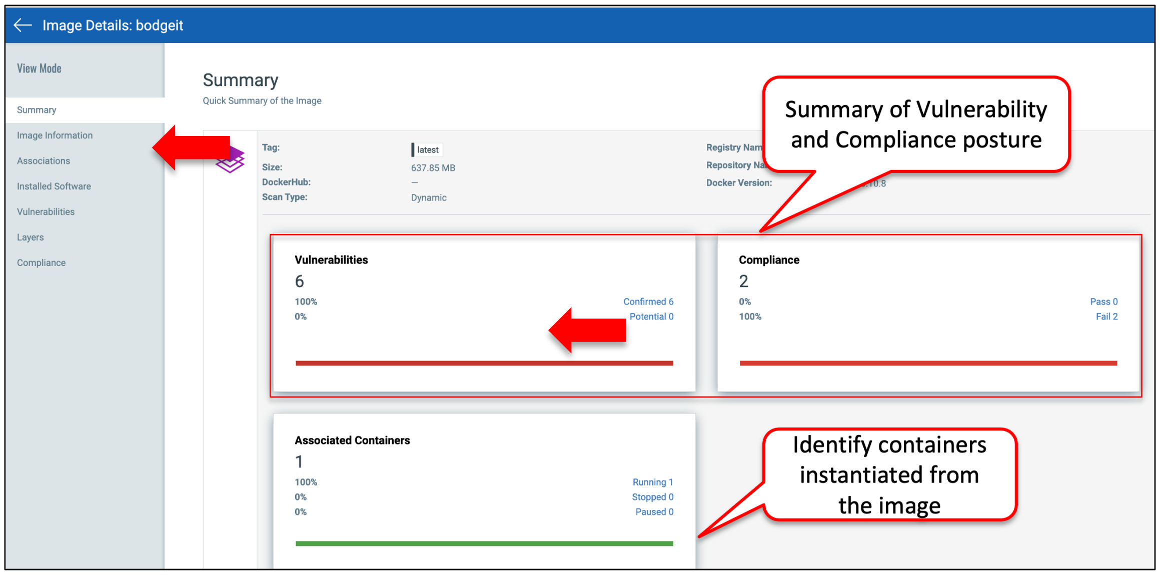
23 Total Images

1 Images detected without CS Sensor | 16 Images with Sev 5, 4 Vulnerabilities | 0 Docker Hub Official Images | 8 Images not Compliant

REGISTRY	
docker.io	22
registry-1.docker....	6
VULNERABILITIES	
Severity 5	33
Severity 4	134
Severity 3	415
Severity 2	25
Severity 1	13
COMPLIANCE POSTURE	
FAIL	15
PASS	1

	POSITORY	CREATED ON		RS	VULNERABILITIES	COMPLIANCE
ker.io	**bodgeit** Image Id: 3b207201ac2e	Aug 17, 2021		sts: 1	6	2
docker.io	**tomcat** Image Id: 710ec5c56683	Aug 07, 2021	latest	0 On Hosts: 1	6	2
docker.io	**ubuntu** Image Id: 1318b700e415	Jul 26, 2021	latest	0 On Hosts: 1	2	2
docker.io	**bodgeit** Image Id: 16852259ef4d	Mar 11, 2021	latest 1 more...	0 On Hosts: 1	9	2

You can use the "Quick Actions" menu on any image to see image and vulnerability details for that image.



Summary

Quick Summary of the Image

Tag:	latest
Size:	637.85 MB
DockerHub:	—
Scan Type:	Dynamic

Registry Nam
Repository Na
Docker Version: .10.8

Vulnerabilities
6
100% Confirmed 6
0% Potential 0

Compliance
2
0% Pass 0
100% Fail 2

Associated Containers
1
100% Running 1
0% Stopped 0
0% Paused 0

In this instance there are multiple confirmed vulnerabilities and compliance control failures detected in the bodgeit image. These come from the Tomcat and bash shell layer included in the image.

Here you can also see the containers(s) presently associated with the image.

***IMPORTANT** – Watch for vulnerable images that are associated with a large number of containers. This is an important consideration when prioritizing the deployment and scheduling of patches.*

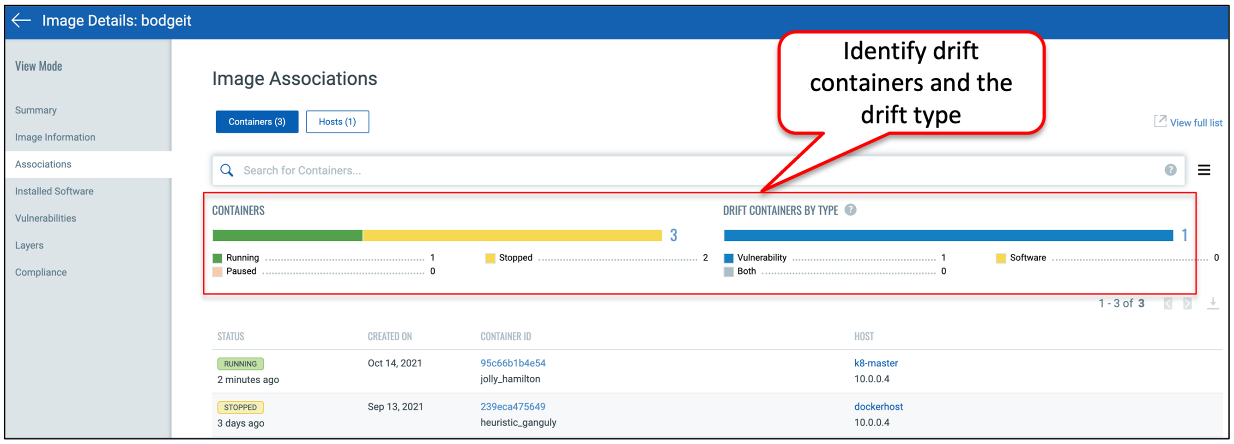


Image Associations

CONTAINERS 3
🟩 Running 1
🟨 Stopped 2
🟧 Paused 0

DRIFT CONTAINERS BY TYPE 1
🟦 Vulnerability 1
🟨 Software 0
⬜ Both 0

1 - 3 of 3

STATUS	CREATED ON	CONTAINER ID	HOST
RUNNING 2 minutes ago	Oct 14, 2021	95c66b1b4e54 jolly_hamilton	k8-master 10.0.0.4
STOPPED 3 days ago	Sep 13, 2021	239eca475649 heuristic_ganguly	dockerhost 10.0.0.4

Any drift containers discovered by the Qualys Container Security application would be displayed here. Drift Containers are those which contain vulnerabilities or software, not found in the image from which the container is spawned. This is typically considered to be abnormal behaviour and may even be an indication of some type of malicious activity that needs to be investigated. Vulnerabilities associated with drift containers are classified as either **New**, **Fixed** or **Varied**:

- **New** – vulnerabilities found in a container that are not present in the image from which the container is spawned.
- **Fixed** - vulnerabilities not found in the container but are present in the parent image.
- **Varied** - vulnerabilities found in both containers and images, but detection varies and is inconsistent between them.

Software associated with drift containers are classified as **new** or **removed**:

- **New** - software found in the Container but not in the image from which the container is spawned.
- **Removed** - software not seen in the Container but is present in the parent Image.

***IMPORTANT** – Identify and investigate drift containers for potential breaches or malicious activity.*

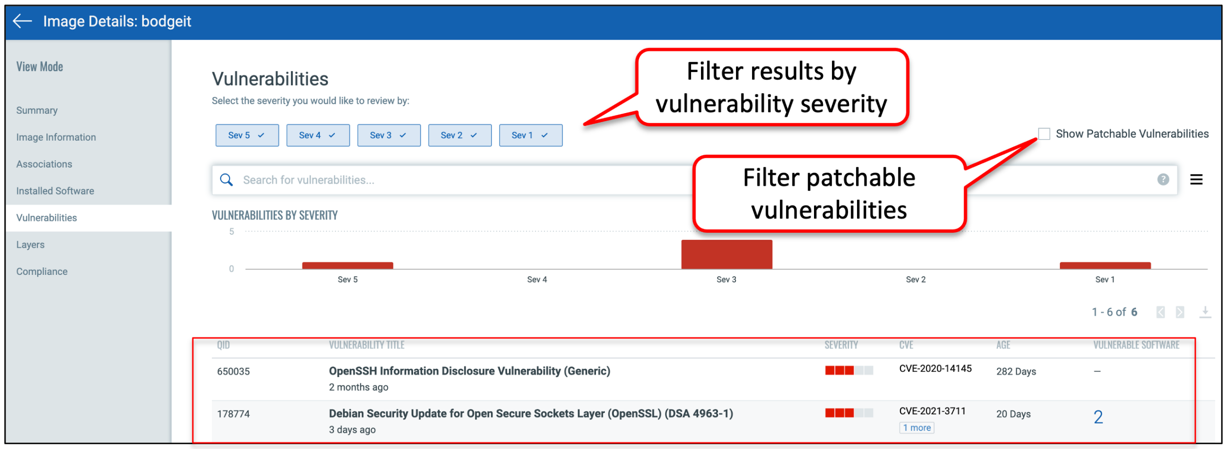


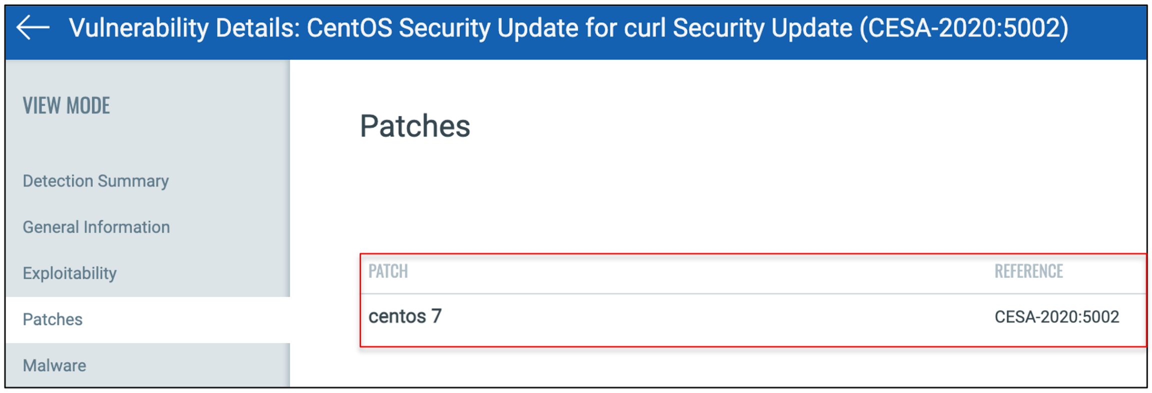
Image Details: bodgeit
View Mode
Summary
Image Information
Associations
Installed Software
Vulnerabilities
Layers
Compliance
Vulnerabilities
Select the severity you would like to review by:
Sev 5 ✓
Sev 4 ✓
Sev 3 ✓
Sev 2 ✓
Sev 1 ✓
Filter results by vulnerability severity
Show Patchable Vulnerabilities
Filter patchable vulnerabilities
Search for vulnerabilities...
VULNERABILITIES BY SEVERITY
5
0
Sev 5
Sev 4
Sev 3
Sev 2
Sev 1
1 - 6 of 6

QID	VULNERABILITY TITLE	SEVERITY	CVE	AGE	VULNERABLE SOFTWARE
650035	**OpenSSH Information Disclosure Vulnerability (Generic)** 2 months ago	🟥🟥🟥⬜⬜	CVE-2020-14145	282 Days	–
178774	**Debian Security Update for Open Secure Sockets Layer (OpenSSL) (DSA 4963-1)** 3 days ago	🟥🟥🟥⬜⬜	CVE-2021-3711 1 more	20 Days	2

Under Vulnerabilities, you can the list of all vulnerability QIDs identified in the image.

Initially all vulnerabilities are listed. You can use the vulnerability severity filters to focus on specific vulnerabilities from the list.

Also selecting the “Show Patchable Vulnerabilities” check box (upper-right corner) filters only those vulnerabilities that have a patch from the vendor.



Vulnerability Details: CentOS Security Update for curl Security Update (CESA-2020:5002)

Patches

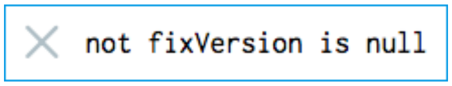
PATCH	REFERENCE
centos 7	CESA-2020:5002

Using the Quick Actions menu for a QID, you can look into vulnerability details which shows information about the patch that is required to address the vulnerability.

You can get a different view of the image vulnerabilities from under the “Installed Software” option.

Initially, all image software applications are listed.

To view only applications with patchable vulnerabilities, you can use the Search field to execute the following query:



```
not fixVersion is null
```

Alternatively, you could simply click the patchable vulnerabilities (within the “TOTAL SOFTWARE” bar graph) to produce the same query.

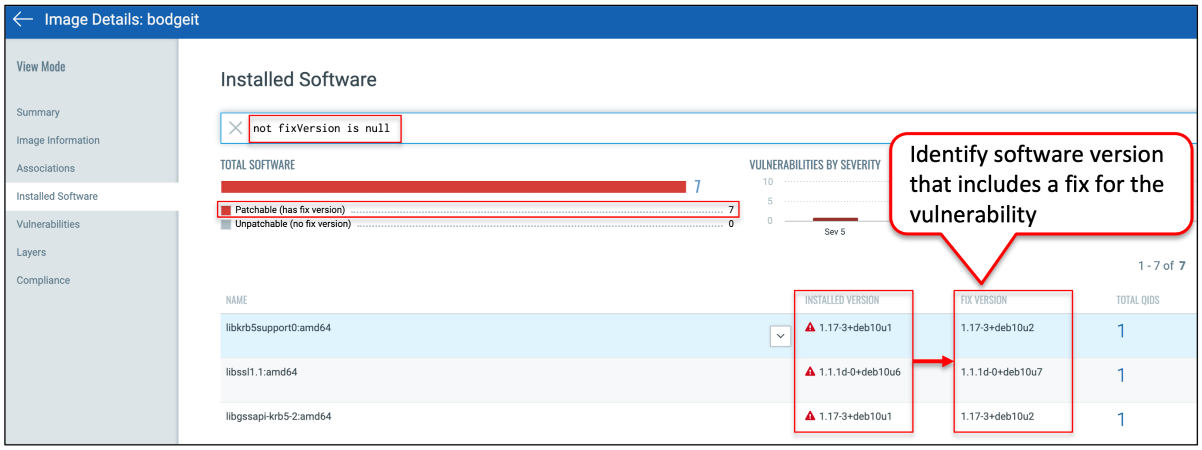


Image Details: bodgeit

View Mode
Summary
Image Information
Associations
Installed Software
Vulnerabilities
Layers
Compliance

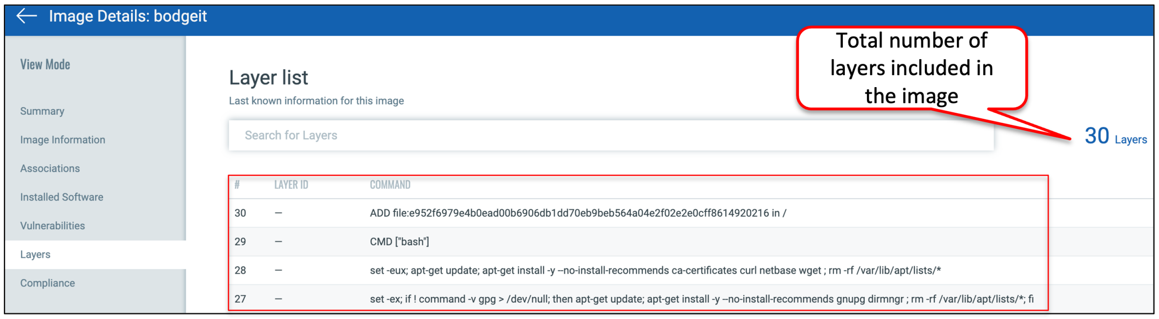
Installed Software

```
not fixVersion is null
```

TOTAL SOFTWARE
7
Patchable (has fix version) 7
Unpatchable (no fix version) 0
VULNERABILITIES BY SEVERITY
10
5
0
Sev 5
Identify software version that includes a fix for the vulnerability
1 - 7 of 7

NAME	INSTALLED VERSION	FIX VERSION	TOTAL QIDS
libkrb5support0:amd64	1.17-3+deb10u1	1.17-3+deb10u2	1
libssl1.1:amd64	1.1.1d-0+deb10u6	1.1.1d-0+deb10u7	1
libgssapi-krb5-2:amd64	1.17-3+deb10u1	1.17-3+deb10u2	1

The currently “installed” version along with its “fix” version is provided for each vulnerable application.

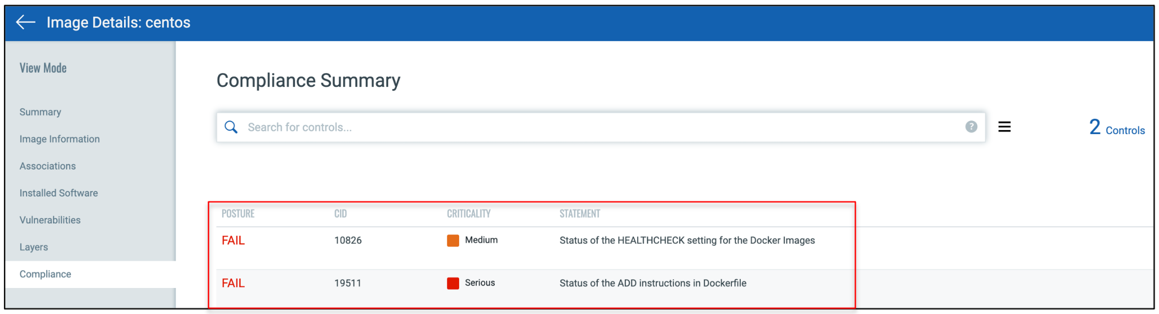


Layer list

Last known information for this image

#	LAYER ID	COMMAND
30	—	ADD file:e952f6979e4b0ead00b6906db1dd70eb9beb564a04e2f02e2e0cff8614920216 in /
29	—	CMD ["bash"]
28	—	set -eux; apt-get update; apt-get install -y --no-install-recommends ca-certificates curl netbase wget ; rm -rf /var/lib/apt/lists/*
27	—	set -ex; if ! command -v gpg > /dev/null; then apt-get update; apt-get install -y --no-install-recommends gnupg dirmngr ; rm -rf /var/lib/apt/lists/*; fi

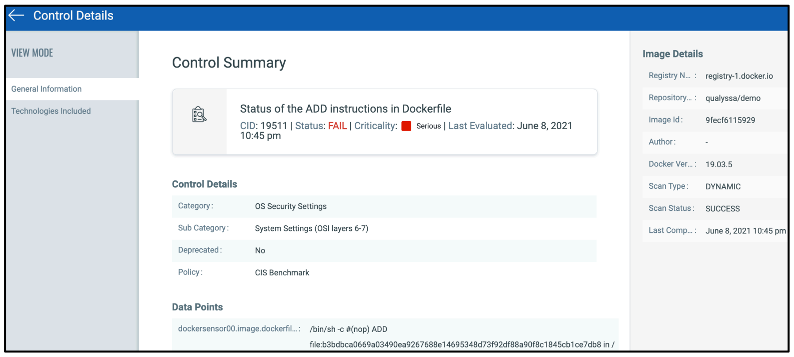
Notice that there are multiple layers in this image. These layers were built from the Docker repository when we built the bodgeit image using the "docker build" command.



Compliance Summary

POSTURE	CID	CRITICALITY	STATEMENT
FAIL	10826	🟧 Medium	Status of the HEALTHCHECK setting for the Docker Images
FAIL	19511	🟥 Serious	Status of the ADD instructions in Dockerfile

Under Compliance, you can see details of the Control IDs (CIDs) that were evaluated for the image and the pass/fail results.



Control Summary

Status of the ADD instructions in Dockerfile

CID: 19511 | Status: FAIL | Criticality: 🟥 Serious | Last Evaluated: June 8, 2021 10:45 pm

Control Details

Category:	OS Security Settings
Sub Category:	System Settings (OSI layers 6-7)
Deprecated:	No
Policy:	CIS Benchmark

Data Points

dockersensor00.image.dockerfil...:	/bin/sh -c #(nop) ADD file:b3bdbca0669a03490ea9267688e14695348d73f92df88a90f8c1845cb1ce7db8 in /

Image Details

Registry N... :	registry-1.docker.io
Repository... :	qualyssa/demo
Image Id :	9fecf6115929
Author :	-
Docker Ver... :	19.03.5
Scan Type :	DYNAMIC
Scan Status :	SUCCESS
Last Comp... :	June 8, 2021 10:45 pm

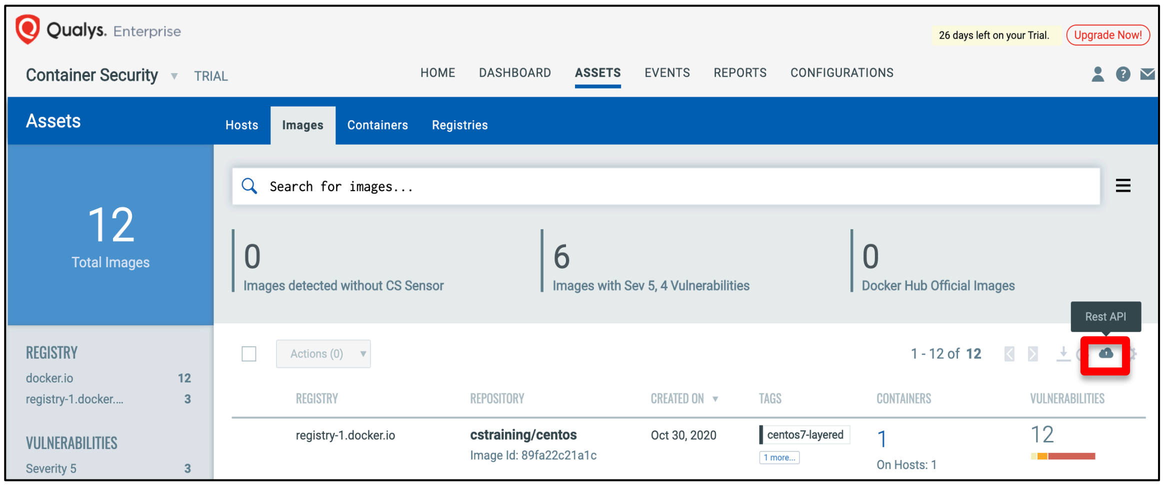
You can drill down into the details for any image or container to see compliance information, including the list of controls that were scanned with control details (CID, criticality, statement, category, technologies and remediation information).

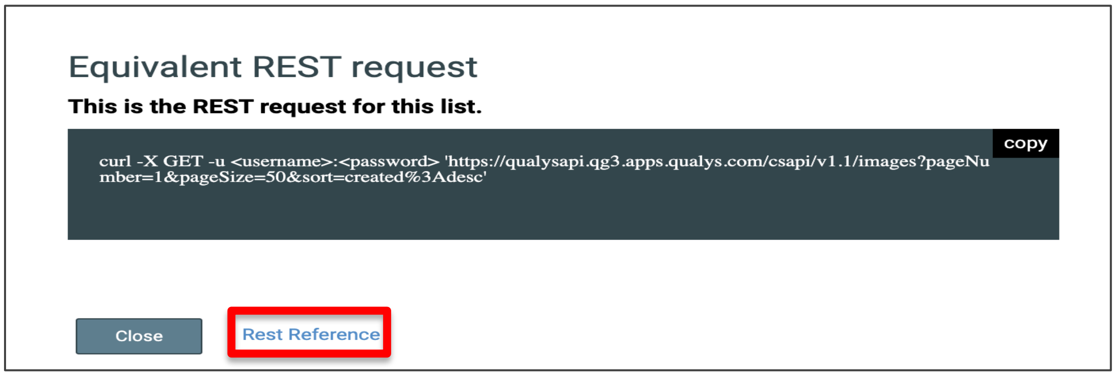
Navigate to the following URL to view the "Assess Containerized Applications" tutorial:

PLAY http://ior.ad/7hbB

Working with the API

All features of Container Security are available through REST APIs. You can quickly generate API calls using the “Rest API” icon in the Qualys UI.





```
curl -X GET -u <username>:<password> 'https://qualysapi.qg3.apps.qualys.com/csapi/v1.1/images?pageNumber=1&pageSize=50&sort=created%3Adesc'
```

In this instance, the REST API function call that was used to produce the displayed list of images or containers is displayed. You can click the “copy” button to reproduce this function call in other application environments. Also, the “Rest Reference” link can be used to view the Swagger UI for building REST API function calls. More API information is available in the Qualys Container Security User Guide: https://www.qualys.com/docs/qualys-container-security-user-guide.pdf.

Dashboards

Tracking the inventory and attack surface is important. Dashboards within the Container Security application, allow you to quickly customize widgets that display:

- Total container and image count
- Containers by state
- Vulnerable images
- Drift containers and more..

You can also build your own custom dashboards and widgets to track the attack surface.

For instance, it will be a serious situation when containers are drifting, and they are running in privileged mode and they have severity 5 type of vulnerabilities. This means the attack surface is very high. The following example query can be used to build such a custom widget which tracks vulnerable containers using a combination of such factors:

```
vulnerabilities.severity:5 and isDrift:true and isRoot:true
```

Navigate to the following URL to view the "Dashboards" tutorial:

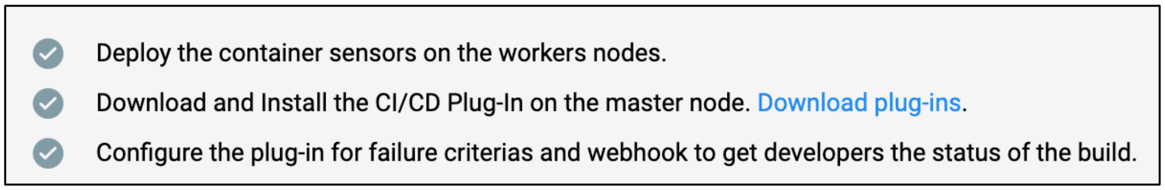
PLAY https://ior.ad/7SDE

Secure the Build Pipeline

Securing the CI/CD pipeline is about providing necessary security tooling to the DevOps teams so that security can be embedded into the build pipeline from the very beginning.

Jenkins is a popular, open source CI/CD automation tool which is used to implement CI/CD workflows, called pipelines. It has a distributed architecture that comprises of the master node and one or more build or worker nodes.

With the Qualys plugin for Jenkins and the Container Sensor, you can secure the Jenkins build pipeline and block images containing high risk vulnerabilities from entering your production environment.



- ✓ Deploy the container sensors on the workers nodes.
- ✓ Download and Install the CI/CD Plug-In on the master node. Download plug-ins.
- ✓ Configure the plug-in for failure criterias and webhook to get developers the status of the build.

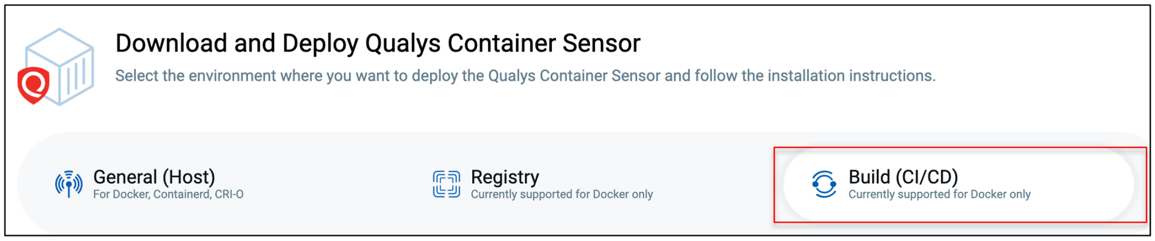
You must perform the following steps to scan images in the Jenkins build pipeline with Qualys Container Security:

1. Deploy the Build (CI/CD) Sensor on all build/worker nodes
2. Deploy the Qualys Plugin on the Jenkins Master
3. Configure Qualys API access and image validation policy
4. Add Qualys scan step to the build pipeline

Download and Install the Sensor Image

The first step is to download and deploy the Build or CI/CD Sensor on each node where images are built. This can be standalone or clustered nodes.

The steps for installing the CI/CD Sensor are similar to that of the General\Registry Sensor. Note that this Sensor currently supports only the Docker runtime.



Download and Deploy Qualys Container Sensor

Select the environment where you want to deploy the Qualys Container Sensor and follow the installation instructions.

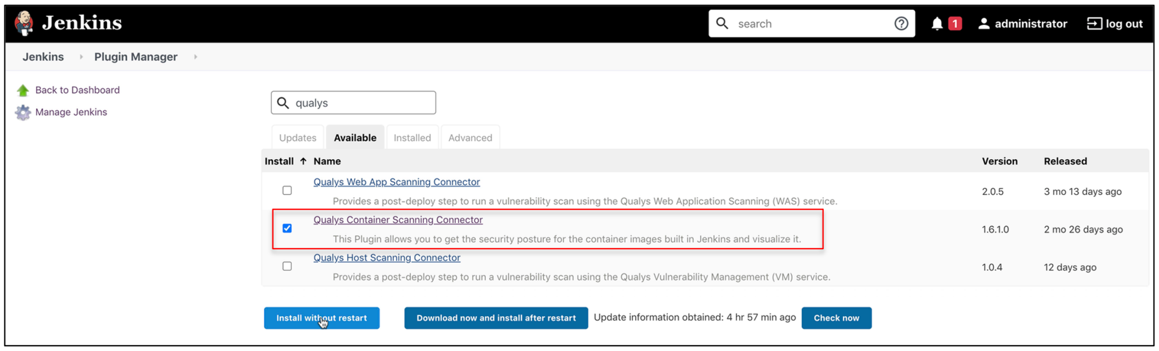
General (Host)
For Docker, Containerd, CRI-O
Registry
Currently supported for Docker only
Build (CI/CD)
Currently supported for Docker only

The CI/CD Sensor is responsible for performing a vulnerability scan of any container image that is built on the build node as defined in the build pipeline.

Install the Qualys Jenkins Plugin

Next, you need to deploy the Qualys plugin\Container Scanning Connector on the Jenkins master. The plugin is available in the Jenkins marketplace.

You can simply search for “Qualys” under Available plugins on the Jenkins master and install the plugin.



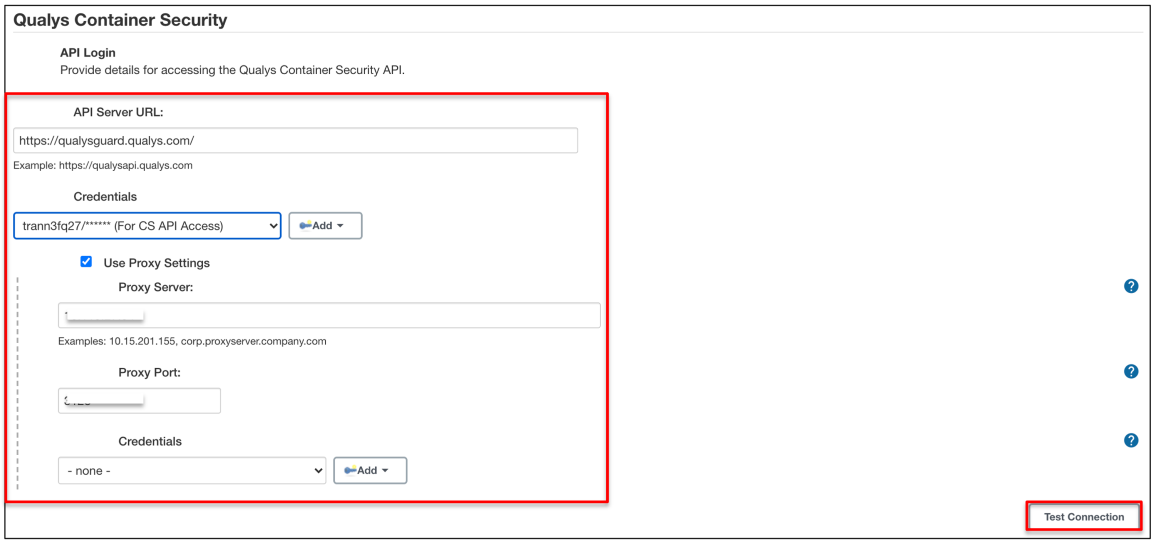
Install ↑	Name	Version	Released
☐	Qualys Web App Scanning Connector Provides a post-deploy step to run a vulnerability scan using the Qualys Web Application Scanning (WAS) service.	2.0.5	3 mo 13 days ago
☑	Qualys Container Scanning Connector This Plugin allows you to get the security posture for the container images built in Jenkins and visualize it.	1.6.1.0	2 mo 26 days ago
☐	Qualys Host Scanning Connector Provides a post-deploy step to run a vulnerability scan using the Qualys Vulnerability Management (VM) service.	1.0.4	12 days ago

Update information obtained: 4 hr 57 min ago

This plugin performs image validation using a policy and ensures that images with high risk vulnerabilities do not make it to the deployment stage.

Configure API Access and Image Validation Policy

The plugin regularly polls the Qualys cloud platform using APIs for vulnerability information of scanned images. You need to provide information such as the Qualys API gateway URL and the Qualys user credentials to access the API endpoint.


Qualys Container Security

API Login

Provide details for accessing the Qualys Container Security API.

API Server URL: https://qualysguard.qualys.com/

Example: https://qualysapi.qualys.com

Credentials: trann3fq27/****** (For CS API Access)

☑ Use Proxy Settings

Proxy Server: [redacted]

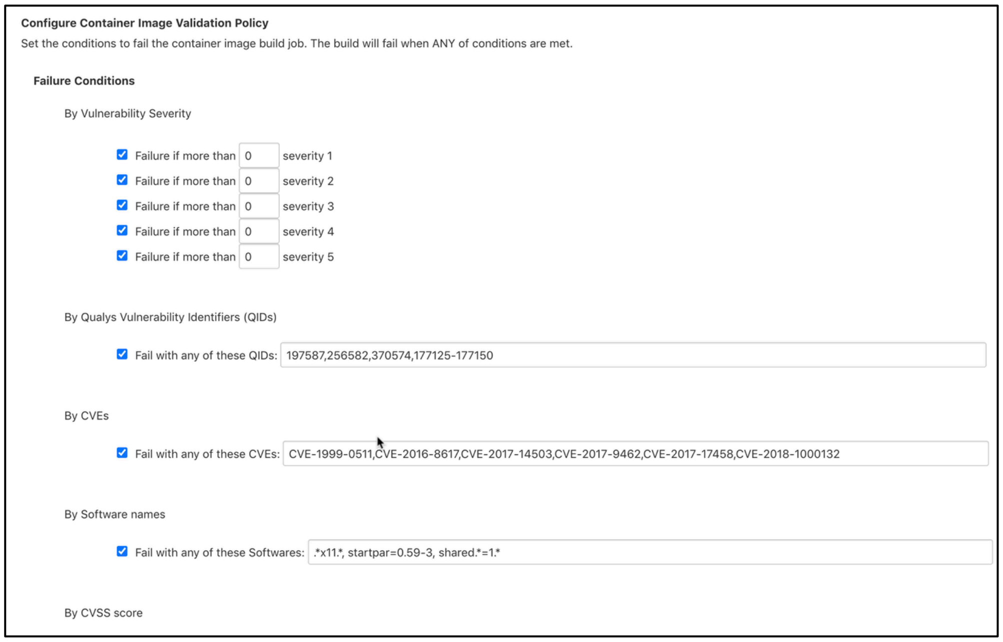
Examples: 10.15.201.155, corp.proxyserver.company.com

Proxy Port: [redacted]

Credentials: - none -

The “Test Connection” button helps you check connectivity to the Qualys platform.

You also need to configure an image validation policy (gate policy) with the build pass/fail criteria, the data collection frequency and image details for validation.



Configure Container Image Validation Policy

Set the conditions to fail the container image build job. The build will fail when ANY of conditions are met.

Failure Conditions

By Vulnerability Severity

- [x] Failure if more than 0 severity 1
- [x] Failure if more than 0 severity 2
- [x] Failure if more than 0 severity 3
- [x] Failure if more than 0 severity 4
- [x] Failure if more than 0 severity 5

By Qualys Vulnerability Identifiers (QIDs)

- [x] Fail with any of these QIDs: 197587,256582,370574,177125-177150

By CVEs

- [x] Fail with any of these CVEs: CVE-1999-0511,CVE-2016-8617,CVE-2017-14503,CVE-2017-9462,CVE-2017-17458,CVE-2018-1000132

By Software names

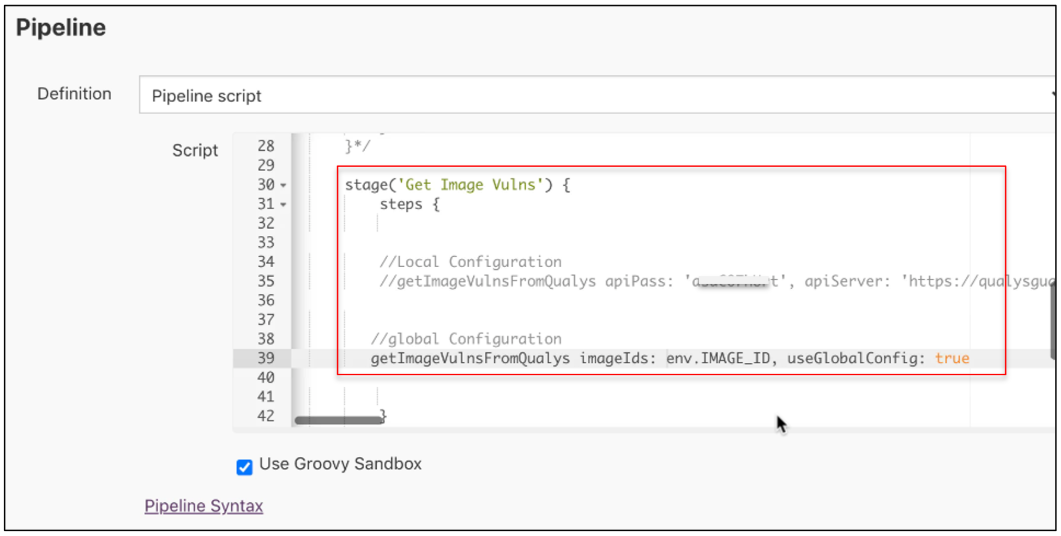
- [x] Fail with any of these Softwares: .*x11.*, startpar=0.59-3, shared.*=1.*

By CVSS score

The policy can be set based on criteria such as vulnerability severity, presence of specific software in the image, QIDs, CVE IDs and CVSS score.

Add the Qualys Scan Task to the Build Pipeline

The next step is to add the Qualys scan task to the build pipeline.



Pipeline

Definition: Pipeline script

Script

```
}*/

stage('Get Image Vulns') {
    steps {


    //Local Configuration
    //getImageVulnsFromQualys apiPass: 'a[redacted]t', apiServer: 'https://qualysgu


   //global Configuration
   getImageVulnsFromQualys imageIds: env.IMAGE_ID, useGlobalConfig: true


    }
```

- [x] Use Groovy Sandbox

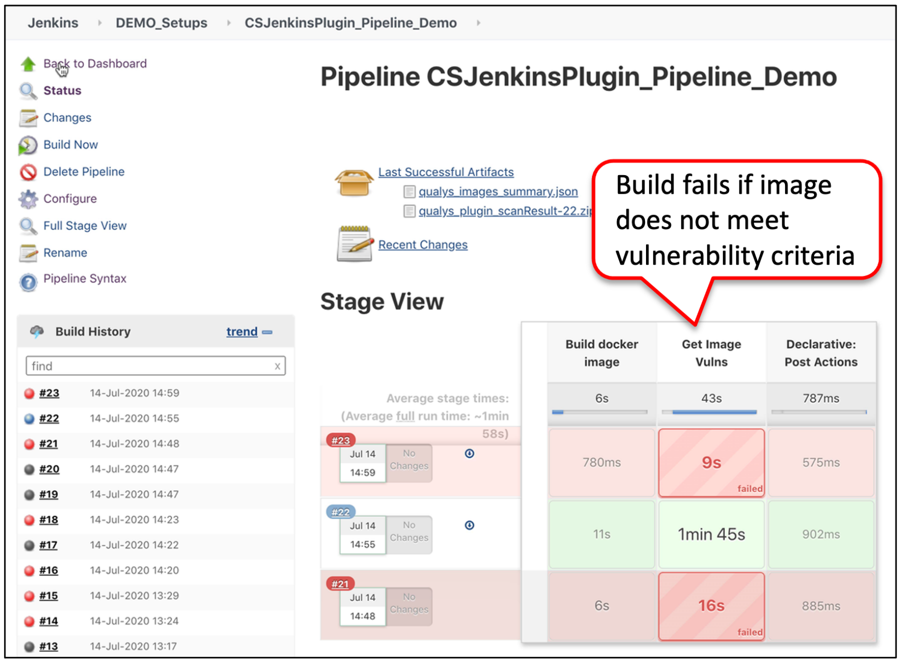
Pipeline Syntax

The Qualys scan task **“getImageVulnsFromQualys”** must run after the image is built and before it is pushed to a registry.

You must also identify the image(s) that will be scanned by Qualys. This can be done using the image ID, image tag or image name or by setting an environment variable. Also, this can be a global configuration or a local (pipeline specific) configuration.

A sample pipeline script with the Qualys scan step is available in the following GitHub repository:
https://github.com/Qualys/community/blob/master/containerSecurity/sample_Jenkinsfile.groovy

When Jenkins executes a build job to package code into a container image, the Qualys plugin tags the resulting image appropriately so that it can be identified as a scan target by the CICD Sensor on the build node.


Jenkins › DEMO_Setups › CSJenkinsPlugin_Pipeline_Demo ›
Back to Dashboard
Status
Changes
Build Now
Delete Pipeline
Configure
Full Stage View
Rename
Pipeline Syntax
Last Successful Artifacts
qualys_images_summary.json
qualys_plugin_scanResult-22.zip
Recent Changes
Build History
trend
find
Build fails if image does not meet vulnerability criteria

Pipeline CSJenkinsPlugin_Pipeline_Demo

#23	14-Jul-2020 14:59
#22	14-Jul-2020 14:55
#21	14-Jul-2020 14:48
#20	14-Jul-2020 14:47
#19	14-Jul-2020 14:47
#18	14-Jul-2020 14:23
#17	14-Jul-2020 14:22
#16	14-Jul-2020 14:20
#15	14-Jul-2020 13:29
#14	14-Jul-2020 13:24
#13	14-Jul-2020 13:17

Stage View

	Build docker image	Get Image Vulns	Declarative: Post Actions
Average stage times: (Average full run time: ~1min 58s)	6s	43s	787ms
#23 Jul 14 14:59 No Changes	780ms	9s failed	575ms
#22 Jul 14 14:55 No Changes	11s	1min 45s	902ms
#21 Jul 14 14:48 No Changes	6s	16s failed	885ms

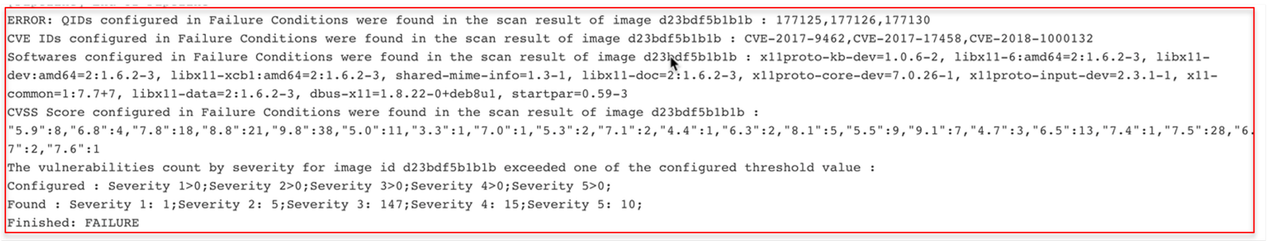
The Container Sensor monitors Docker events on the build node. As soon as the image is built, the CI/CD Sensor scans the target image and uploads the results to the Qualys cloud platform for processing.

The Qualys plugin then pulls the scan results from the Qualys cloud platform using APIs and passes or fails the build based on scan results.

After the build job completes you can review all activities pertaining to the job on the Jenkins master.

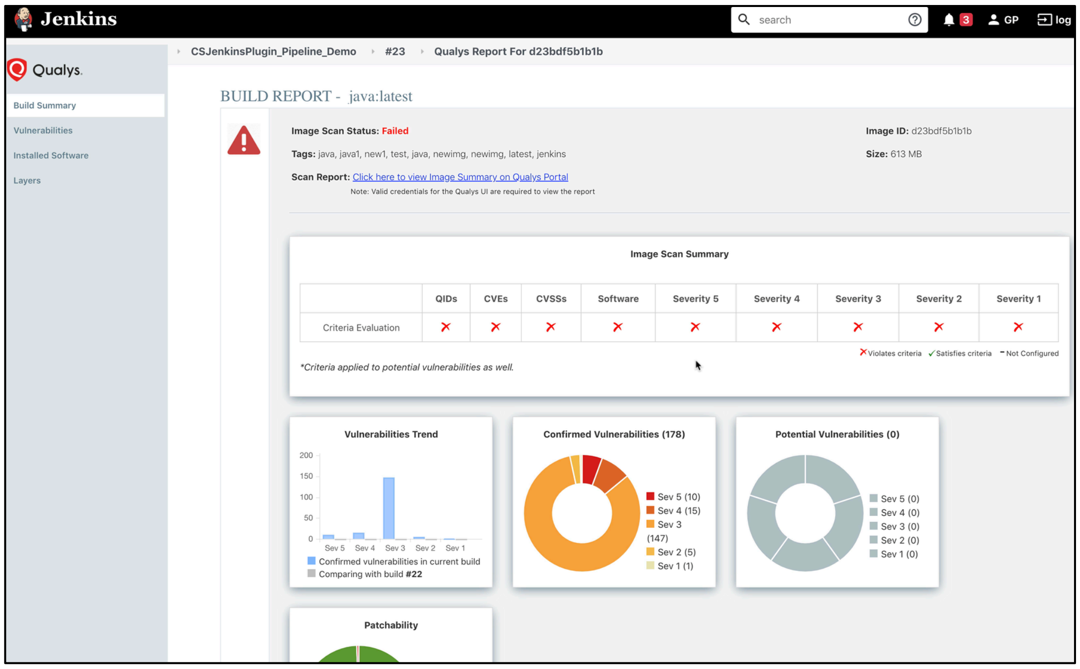
View Scan Results

The build job fails if the image does not meet the vulnerability criteria defined in the plugin configuration.



```
ERROR: QIDs configured in Failure Conditions were found in the scan result of image d23bdf5b1b1b : 177125,177126,177130
CVE IDs configured in Failure Conditions were found in the scan result of image d23bdf5b1b1b : CVE-2017-9462,CVE-2017-17458,CVE-2018-1000132
Softwares configured in Failure Conditions were found in the scan result of image d23bdf5b1b1b : x11proto-kb-dev=1.0.6-2, libx11-6:amd64=2:1.6.2-3, libx11-
dev:amd64=2:1.6.2-3, libx11-xcb1:amd64=2:1.6.2-3, shared-mime-info=1.3-1, libx11-doc=2:1.6.2-3, x11proto-core-dev=7.0.26-1, x11proto-input-dev=2.3.1-1, x11-
common=1:7.7+7, libx11-data=2:1.6.2-3, dbus-x11=1.8.22-0+deb8u1, startpar=0.59-3
CVSS Score configured in Failure Conditions were found in the scan result of image d23bdf5b1b1b :
"5.9":8,"6.8":4,"7.8":18,"8.8":21,"9.8":38,"5.0":11,"3.3":1,"7.0":1,"5.3":2,"7.1":2,"4.4":1,"6.3":2,"8.1":5,"5.5":9,"9.1":7,"4.7":3,"6.5":13,"7.4":1,"7.5":28,"6.
7":2,"7.6":1
The vulnerabilities count by severity for image id d23bdf5b1b1b exceeded one of the configured threshold value :
Configured : Severity 1>0;Severity 2>0;Severity 3>0;Severity 4>0;Severity 5>0;
Found : Severity 1: 1;Severity 2: 5;Severity 3: 147;Severity 4: 15;Severity 5: 10;
Finished: FAILURE
```

The "Console Output" section of the build job provides information about a build failure.



BUILD REPORT - java:latest

Image Scan Status: Failed

Tags: java, java1, new1, test, java, newimg, newimg, latest, jenkins

Scan Report: Click here to view Image Summary on Qualys Portal

Note: Valid credentials for the Qualys UI are required to view the report

Image ID: d23bdf5b1b1b

Size: 613 MB

Image Scan Summary

	QIDs	CVEs	CVSSs	Software	Severity 5	Severity 4	Severity 3	Severity 2	Severity 1
Criteria Evaluation	✗	✗	✗	✗	✗	✗	✗	✗	✗

✗ Violates criteria ✓ Satisfies criteria - Not Configured

Criteria applied to potential vulnerabilities as well.

The Qualys plugin generates a scan report for each container image in the build pipeline which provides details of all vulnerabilities identified in the image along with patch availability and software version that includes a fix for the vulnerability.

Navigate to the following URL to view the "Securing the Jenkins Build Pipeline" tutorial:

PLAY http://ior.ad/7bVW

Secure the Registry

A successful Vulnerability Management program in a containerized environment must also include scanning for vulnerable images in the container registry and practicing good registry hygiene. Because registries are typically trusted as a source of valid, approved software, compromise of an image in a registry can potentially lead to compromise of downstream containers and hosts. So, it is important to ensure that only compliant and secure images make to the registry.
When we scan images during the build phase, we have the metadata which identifies that the image was scanned at the build phase. So, when an image makes it to the registry, we can easily verify if it was scanned in the build pipeline. This information can then be used to ensure that unsecured and non-compliant images do not make it to production.
Also over time the set of images stored in registries can include out-of-date versions. They increase the likelihood of accidental deployment of a known-vulnerable version. So registry scanning provides the necessary visibility that helps maintain proper registry hygiene and ensures that only compliant and secure images are retained in the registry.

Qualys supports scanning images in public and private registries such as Docker Hub, ECR, ACR, GCR, Nexus, jFrog Artifactory and others.

You must perform the following steps to scan images in a Docker Hub registry:

1. Install Registry Sensor on Docker host
2. Add Registry information to the Qualys Container Security application
3. Configure scan job(s)

Registry Scanning Host

It's recommended to setup a dedicated host for registry scanning to avoid resource contention issues. The host must have a docker engine and the Registry Sensor installed on the host needs access to the docker runtime. The host must have a minimum of 20 GB free space on the partition where docker is installed. This is required to scan registry images. Additionally, 1 GB of free space is required for persistent storage. If the repository has a lot of images to scan, the overall scanning time might be longer than usual. You can setup multiple registry scanning hosts to distribute the scanning payload, to reduce the scan time and view the results faster.

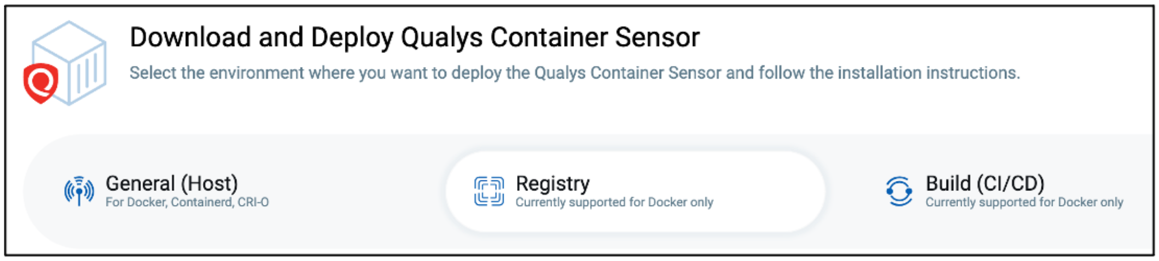
The scan host:

1. Must have access to the target registry.

2. Must have access to the Qualys cloud platform. Communication through proxies is supported.

Install the Registry Sensor

The first step is to download and install the Qualys Registry Sensor on the host(s) designated for scanning images in your registry. This can be on standalone or clustered nodes.


Download and Deploy Qualys Container Sensor

Select the environment where you want to deploy the Qualys Container Sensor and follow the installation instructions.

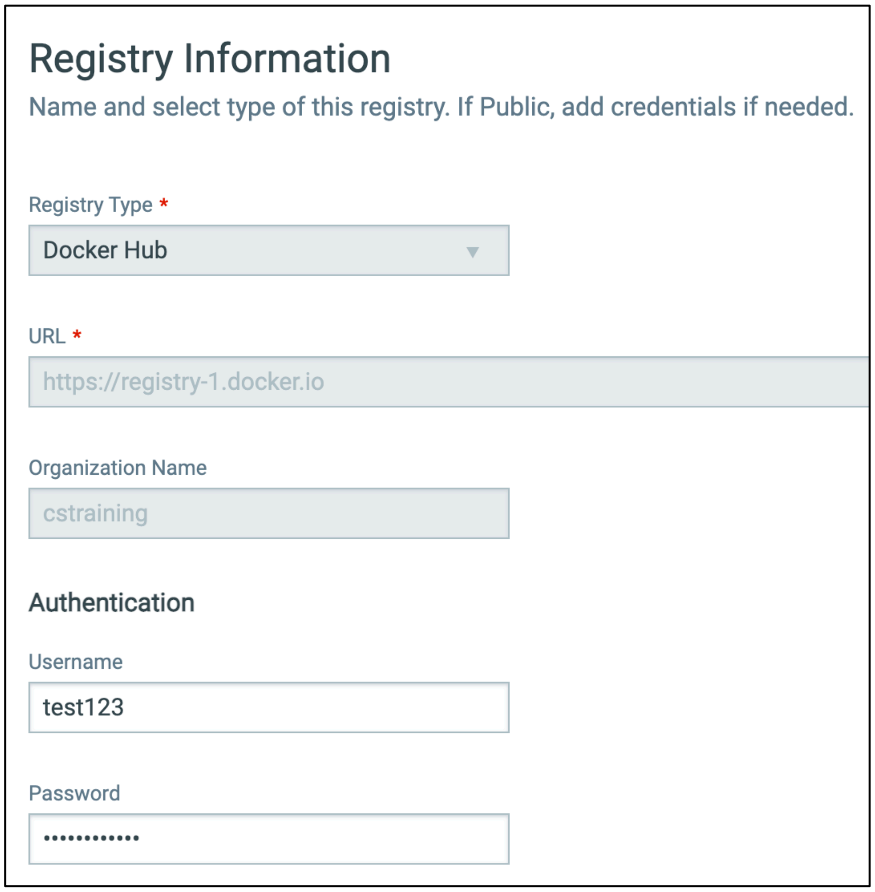
General (Host)
For Docker, Containerd, CRI-O
Registry
Currently supported for Docker only
Build (CI/CD)
Currently supported for Docker only

The steps for installing the Registry Sensor are similar to that of the General\CICD Sensor. Note that this Sensor currently supports only the Docker runtime.

The Registry Sensor is responsible for listing images and performing a vulnerability and compliance scan of identified images that reside in the target registry that is added to the Qualys Container Security application.

Add Registry

The next step is to add the registry details to your Qualys account. This step is performed in the Container Security application and requires your registry location and user credentials for scanning.


Registry Information

Name and select type of this registry. If Public, add credentials if needed.

Registry Type *

Docker Hub

URL *

https://registry-1.docker.io

Organization Name

cstraining

Authentication

Username

test123

Password

••••••••••••

For Docker Hub, the URL path is automatically selected. You need to provide the Organization Name and user credentials to authenticate to the registry.

We recommend using an account that has read only access to the registry for scanning.

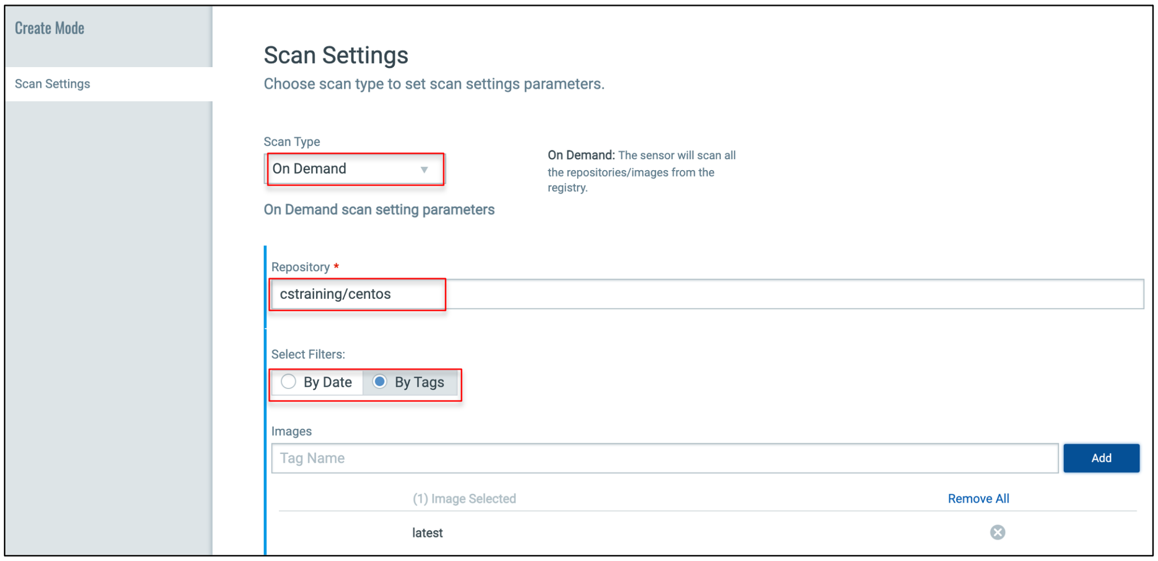
However if you also intend to instrument or protect images in the registry using Qualys Container Runtime Security, you will need to use an account that has both read and write privilege to the registry.

Container Runtime Security and the instrumentation process are discussed in greater detail in the next topic.

Configure Scan Jobs

After adding registry details, you need to configure a scan job(s) for scanning images in the target registry.

You can scan immediately (On Demand scan) or on an on-going basis (Automatic scan).


Create Mode

Scan Settings

Scan Settings

Choose scan type to set scan settings parameters.

Scan Type: On Demand

On Demand: The sensor will scan all the repositories/images from the registry.

On Demand scan setting parameters

Repository *: cstraining/centos

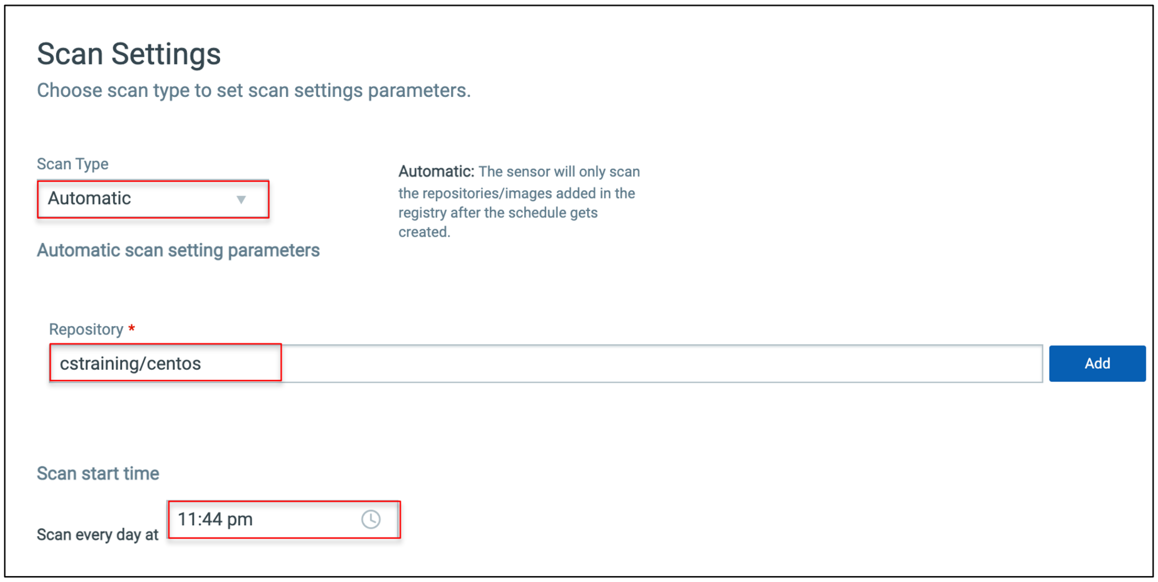
Select Filters: ○ By Date ● By Tags

Images: Tag Name Add

(1) Image Selected — Remove All

latest

An On Demand scan allows you to scan repositories as well as specific images within those repositories “By Date” or “By Tags”. This job will create the initial baseline for scanning.


Scan Settings

Choose scan type to set scan settings parameters.

Scan Type: Automatic

Automatic: The sensor will only scan the repositories/images added in the registry after the schedule gets created.

Automatic scan setting parameters

Repository *: cstraining/centos Add

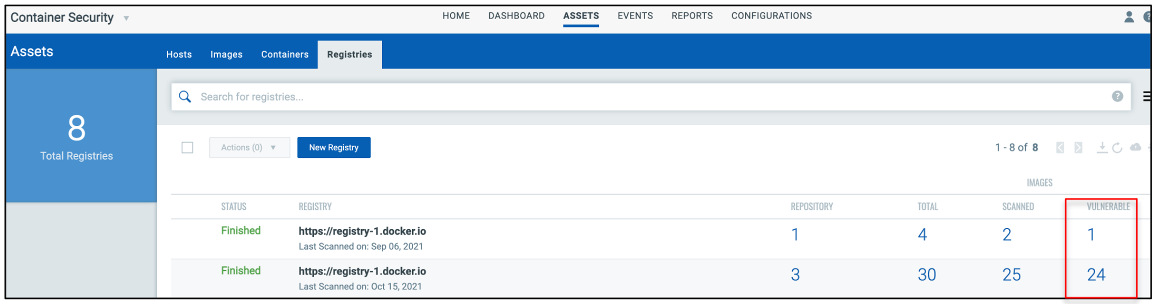
Scan start time

Scan every day at 11:44 pm

With Automatic scan, you can scan entire repositories at a set time every day. As automatic scans are incremental, only the new images that are added to the configured repository since the last scan will be considered for scanning.

View Scan Results

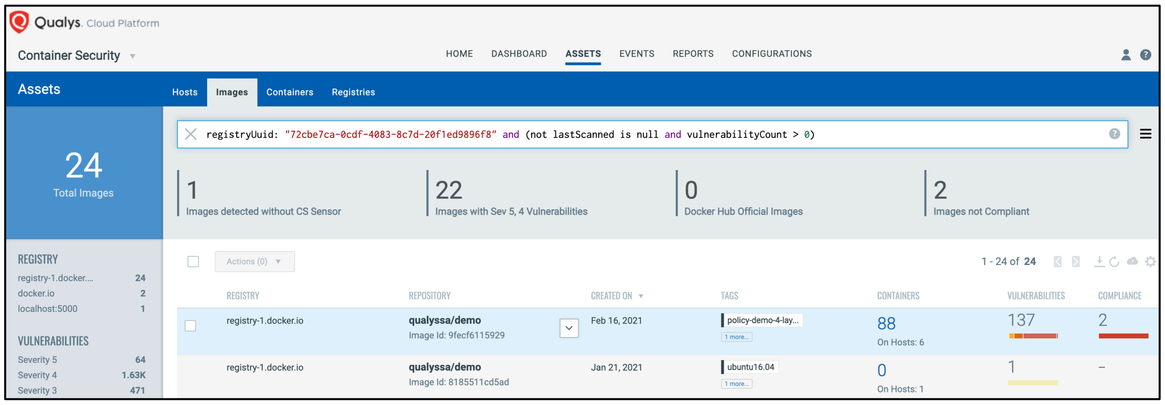
After the scan job completes, you can see the total number of images present in the registry, the number of scanned images and the number of vulnerable images.



8
Total Registries

STATUS	REGISTRY	REPOSITORY	IMAGES TOTAL	IMAGES SCANNED	IMAGES VULNERABLE
Finished	**https://registry-1.docker.io** Last Scanned on: Sep 06, 2021	1	4	2	1
Finished	**https://registry-1.docker.io** Last Scanned on: Oct 15, 2021	3	30	25	24

Clicking the number under the "Vulnerable" column will display information about the vulnerable images in the image repository.



```
registryUuid: "72cbe7ca-0cdf-4083-8c7d-20f1ed9896f8" and (not lastScanned is null and vulnerabilityCount > 0)
```

24
Total Images

1 Images detected without CS Sensor
22 Images with Sev 5, 4 Vulnerabilities
0 Docker Hub Official Images
2 Images not Compliant

REGISTRY	
registry-1.docker....	24
docker.io	2
localhost:5000	1

VULNERABILITIES	
Severity 5	64
Severity 4	1.63K
Severity 3	471

REGISTRY	REPOSITORY	CREATED ON	TAGS	CONTAINERS	VULNERABILITIES	COMPLIANCE
registry-1.docker.io	**qualyssa/demo** Image Id: 9fecf6115929	Feb 16, 2021	policy-demo-4-lay... 1 more...	88 On Hosts: 6	137	2
registry-1.docker.io	**qualyssa/demo** Image Id: 8185511cd5ad	Jan 21, 2021	ubuntu16.04 1 more...	0 On Hosts: 1	1	–

You can then use the quick actions menu to drill down into vulnerabilities or compliance gaps in each image.

Navigate to the following URL to view the "Scan Images in Docker Hub" tutorial:

PLAY http://ior.ad/7ceg

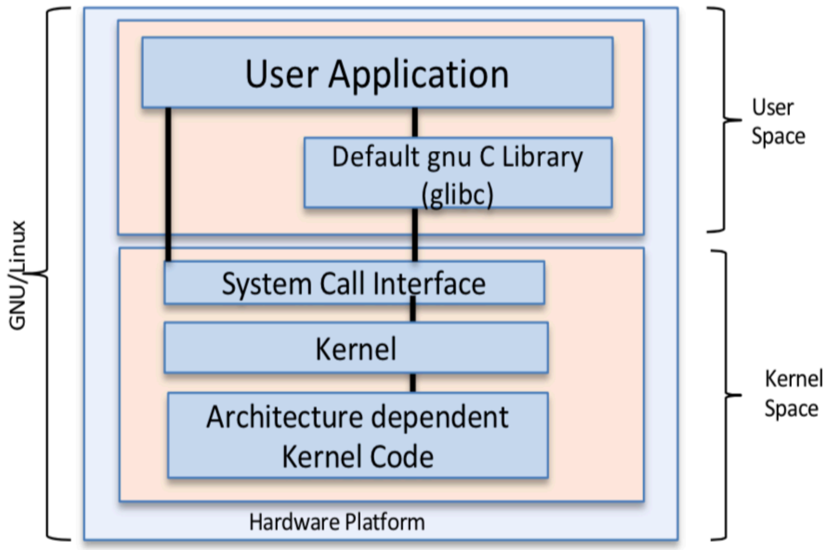
Operationalize Container Runtime Security

Protecting containers with Qualys Container Runtime Security requires instrumenting the container image with the Qualys security layer. We simply drop a few binaries into the image as the security layer.

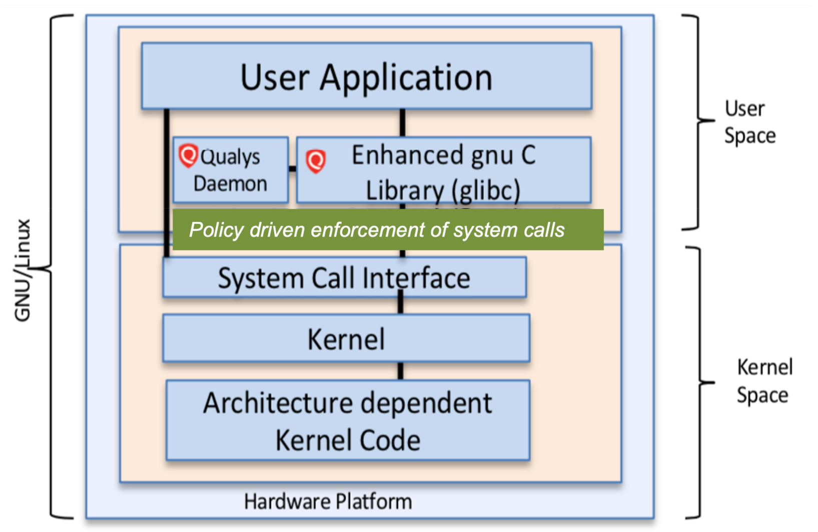
User programs (like text editors, terminal, ssh daemon, etc) interact with the kernel so that the kernel can perform a set of operations on behalf of your user programs that they can't perform themselves.

For example, if a user program needs to do some sort of input/output (IO) operation (open, read, write, etc.) on a file or modify its address space (mmap, sbrk, etc.) it must trigger the kernel to run to complete those actions on its behalf.

These user programs use the GNU C Library, commonly known as glibc, as an intermediary to make system calls (syscalls) to the kernel. System call provides the services of the operating system to user programs via Application Program Interface (API).



Pre-Instrumentation Container


User Application
Qualys Daemon
Enhanced gnu C Library (glibc)
Policy driven enforcement of system calls
System Call Interface
Kernel
Architecture dependent Kernel Code
Hardware Platform
GNU/Linux
User Space
Kernel Space

Container with Qualys Instrumentation

When you instrument an image with Qualys Container Runtime Security, the default glibc library is replaced by the enhanced Qualys glibc library. This layer provides the required visibility and protection for the container in the runtime environment.

Instrumentation Methods

You can instrument images using a **CLI script** (instrumenter.sh) script provided by Qualys or by using the **Instrumenter service**, which is a lightweight microservice that runs as a container in your environment.
The CLI script and the Instrumenter service image are available on https://github.com/Qualys/qualys_crs_instrumenter

Regardless of the option you chose for instrumenting images, the following information is required:

- Qualys API user credentials
- Qualys API gateway URL
- Docker URL
- Proxy server information (required if using a proxy)
- Vault Key, vault address, vault engine, vault path (required if using a vault)

Please consult https://www.qualys.com/platform-identification/ for more information on identifying the Qualys API gateway URL for your subscription.

Verify Image for Instrumentation Support

After setting up the instrumenter CLI script or installing the instrumenter service, you can start instrumenting images to add the Container Runtime Security components to an image.

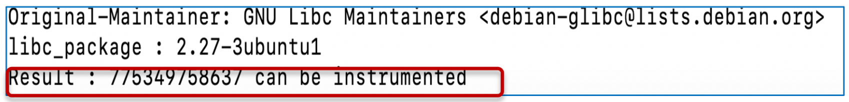
CRS supports specific versions of glibc in an image for instrumentation. Qualys provides a script (check_if_image_instrumentable.sh) to verify if an image can be instrumented.

The script is available on the following GitHub link for download:
https://github.com/Qualys/qualys_crs_instrumenter/commit/377345da9175860e57334bf1045e1336d61a95ab

The following command will verify if the given image supports instrumentation:

```
$ ./check_if_image_instrumentable.sh 7e6257c9f8d8
```

where “`7e6257c9f8d8`” is the “Image ID” of the image to instrument.



```
Original-Maintainer: GNU Libc Maintainers <debian-glibc@lists.debian.org>
libc_package : 2.27-3ubuntu1
Result : 775349758637 can be instrumented
```

You should see a message as illustrated above, which indicates that the image can be instrumented by Qualys.

Instrument Images from the CLI

You can use the CLI mode option to instrument any image on your local host directly or by adding the script as a step in the CI pipeline without the need for a registry scan first. You can edit the **instrumenter.sh** script to configure user specific details for proxy and vault usage.

When done, run the docker CLI script with the minimum required parameters.
Example:

```
./instrumenter.sh --endpoint "qualys_user:my-
password@gateway.qg1.apps.qualys.com/crs/v1.2" --image
"6d9ae1a5c970" [--policyid "5fd20b4321dabf0001fdc464"]
```

Required fields are endpoint and image ID. Policy ID is optional.

<endpoint> is in the format of **username:password@url** if you are not using a vault.
Otherwise only the gateway URL is needed when you are using a vault.
The gateway URL is specific to your Qualys subscription.

Note: The Instrumenter image is stored in a private Docker Hub repository and is automatically downloaded as a part of the CLI script execution. You will need permission to be able to download the image. Please contact Qualys support or reach out to your Qualys TAM to assign the necessary permission to your Docker Hub account to access the Qualys CLI Instrumenter Docker image.

The image to instrument must be present locally where you're running the CLI command.

One command will instrument one image only, and then it will exit as soon as the instrumentation is done.

When you instrument an image using this option, we'll immediately add in our solution and create the instrumented image at the same location and which has the term "-layered" appended to the tag.

The instrumented image will appear in the Container Security application where you can view details about it.

Navigate to the following URL to view the "Instrument Images using CLI" tutorial:

PLAY https://ior.ad/7Ska

Instrumenter Service Deployment

The Instrumenter service is packaged and distributed as a Docker image and runs as a container on a Docker host.

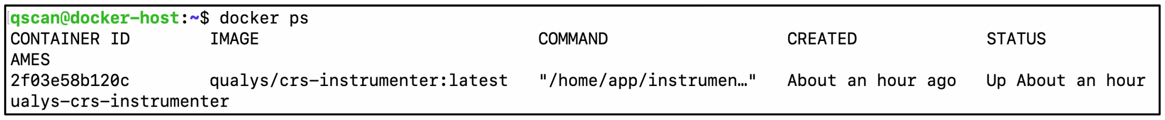
You can deploy the Instrumenter service using any of the following three methods:

- Run the Instrumenter service container using a CLI based command (**instrumenter.sh)**
- Configure and run the Instrumenter service from a docker compose file (**deploy-instrumenter-docker-compose.yml**)
- Configure a Kubernetes template (**instrumenter.yml**) and create a daemonset for the instrumenter.

Notes:

- Only one Instrumenter service per docker host is supported
- Multiple Instrumenter service containers can be deployed considering number of images to be instrumented

- Currently there is no visibility of the Instrumenter Service via CS user interface or the API
- The Qualys cloud platform federates instrumentation requests to the Instrumenter service and instrumentation jobs are delivered to any authenticated Instrumenter service in your environment

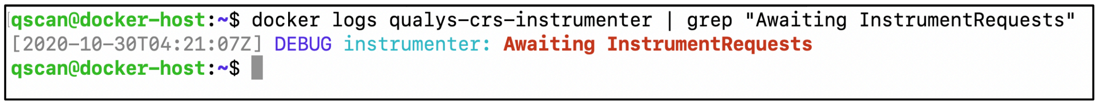


```
qscan@docker-host:~$ docker ps
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATUS
AMES
2f03e58b120c        qualys/crs-instrumenter:latest      "/home/app/instrumen…"   About an hour ago   Up About an hour
ualys-crs-instrumenter
```

After the instrumenter is deployed, you should see it listed as a container in the output of the “`docker ps`” command as illustrated above.

Further, you can check the instrumenter logs to verify the instrumenter is online and functional using this command:

```
docker logs instrumenter | grep "Awaiting
InstrumentRequests"
```



```
qscan@docker-host:~$ docker logs qualys-crs-instrumenter | grep "Awaiting InstrumentRequests"
[2020-10-30T04:21:07Z] DEBUG instrumenter: Awaiting InstrumentRequests
qscan@docker-host:~$
```

If the Instrumenter service was deployed successfully, the output should indicate the status as illustrated above.

Please consult the Qualys Container Runtime Security User Guide for more information on deploying the Instrumenter service
https://www.qualys.com/docs/qualys-container-runtime-security-user-guide.pdf

Navigate to the following URL to view the “Deploy the Instrumenter Service” tutorial:

PLAY https://ior.ad/7Sk5

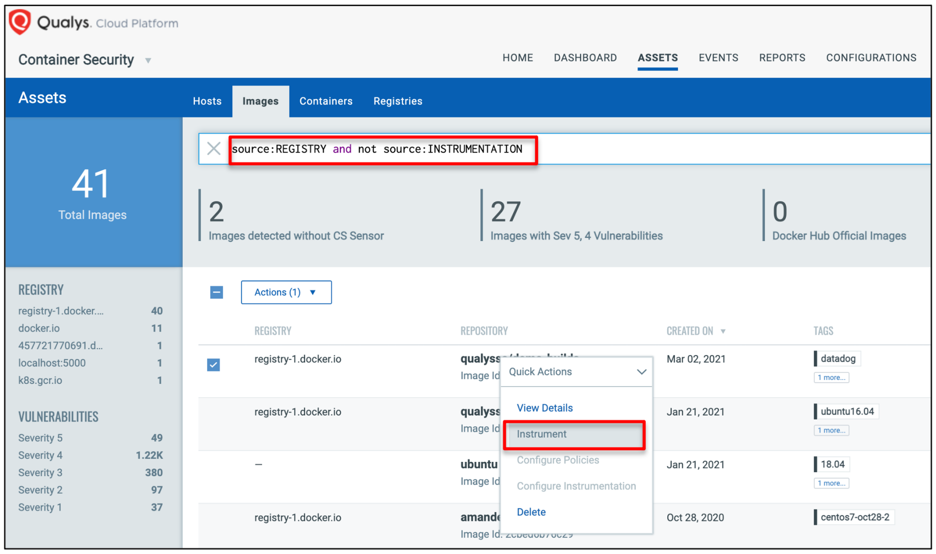
Instrument Images from the UI

The Instrument option in the Container security user interface (UI) lets you instrument container images that have been scanned by the Registry Sensor. The image must also reside in a supported registry (Docker Hub, jFrog Artifactory, Nexus).

You can perform a search under **Assets -> Images** tab in the Container Security application to find images you can instrument.

The following search query will identify images that have been scanned by the Registry Sensor but not yet instrumented:
`source: REGISTRY and not source: INSTRUMENTATION`

You can add additional search fields to help narrow down the list further.



```
source:REGISTRY and not source:INSTRUMENTATION
```

41 Total Images

2 Images detected without CS Sensor

27 Images with Sev 5, 4 Vulnerabilities

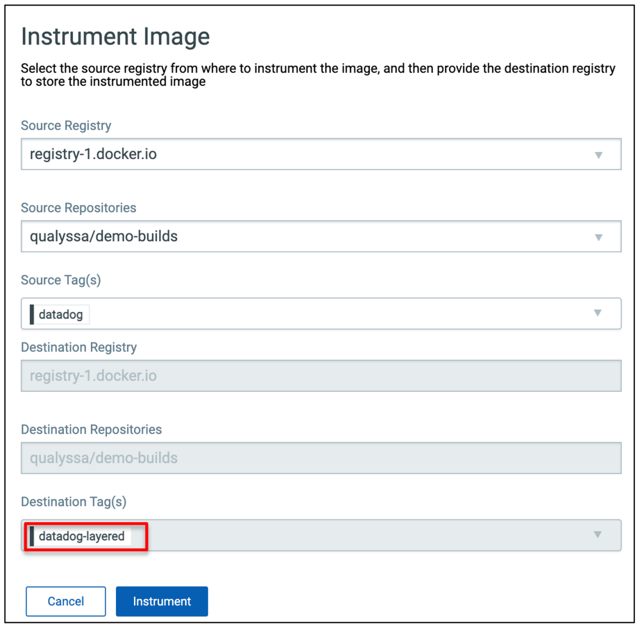
0 Docker Hub Official Images

REGISTRY	
registry-1.docker....	40
docker.io	11
457721770691.d...	1
localhost:5000	1
k8s.gcr.io	1

VULNERABILITIES	
Severity 5	49
Severity 4	1.22K
Severity 3	380
Severity 2	97
Severity 1	37

	REGISTRY	REPOSITORY	CREATED ON	TAGS
✓	registry-1.docker.io	qualyss Image Id	Mar 02, 2021	datadog 1 more...
	registry-1.docker.io	qualyss Image Id	Jan 21, 2021	ubuntu16.04 1 more...
	–	ubuntu Image Id	Jan 21, 2021	18.04 1 more...
	registry-1.docker.io	amande Image Id	Oct 28, 2020	centos7-oct28-2

Then, in the search results you can identify the image you want to instrument and pick **Instrument** from the Quick Actions menu.



Instrument Image

Select the source registry from where to instrument the image, and then provide the destination registry to store the instrumented image

Source Registry: registry-1.docker.io

Source Repositories: qualyssa/demo-builds

Source Tag(s): datadog

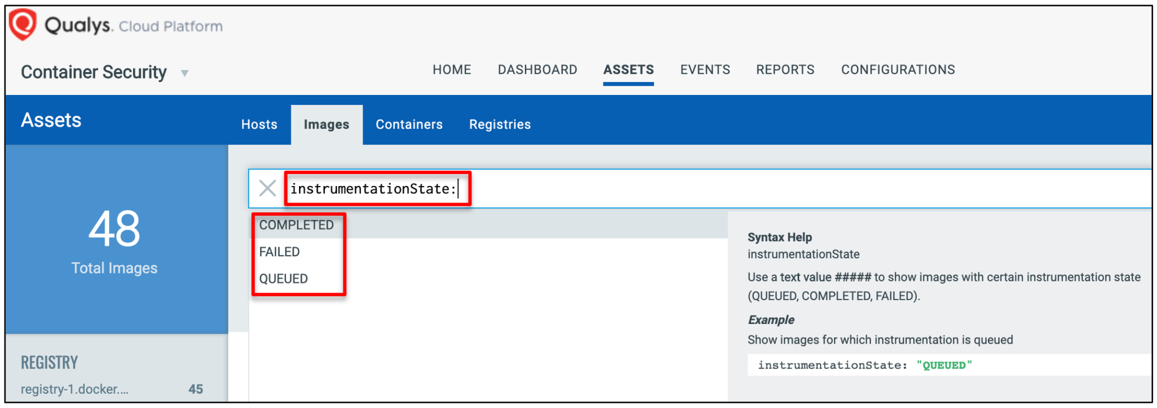
Destination Registry: registry-1.docker.io

Destination Repositories: qualyssa/demo-builds

Destination Tag(s): datadog-layered

We take the source tag and append -layered to create the destination tag. For example, if the source tag is java01 then the destination tag will be java01-layered.

The instrumenter service will pull the image down, add in the Qualys security layer and push the image back to the destination registry.



```
instrumentationState:
```

Syntax Help
instrumentationState
Use a text value ##### to show images with certain instrumentation state (QUEUED, COMPLETED, FAILED).
Example
Show images for which instrumentation is queued

```
instrumentationState: "QUEUED"
```

The following queries will help you identify the status of the instrumentation job in the Container Security application:

- To list Completed jobs: `instrumentationState:COMPLETED`
- To list Queued jobs: `instrumentationState:QUEUED`
- To list Failed jobs: `instrumentationState:FAILED`

If a job fails, you can view instrumenter logs on the Docker host where the Instrumenter service is deployed using the following command:

```
docker logs instrumenter
```

Where *instrumenter* represents the *Container ID* for the Instrumenter service container.

Please contact Qualys support for further assistance to work on any issues concerning failed instrumentation jobs.

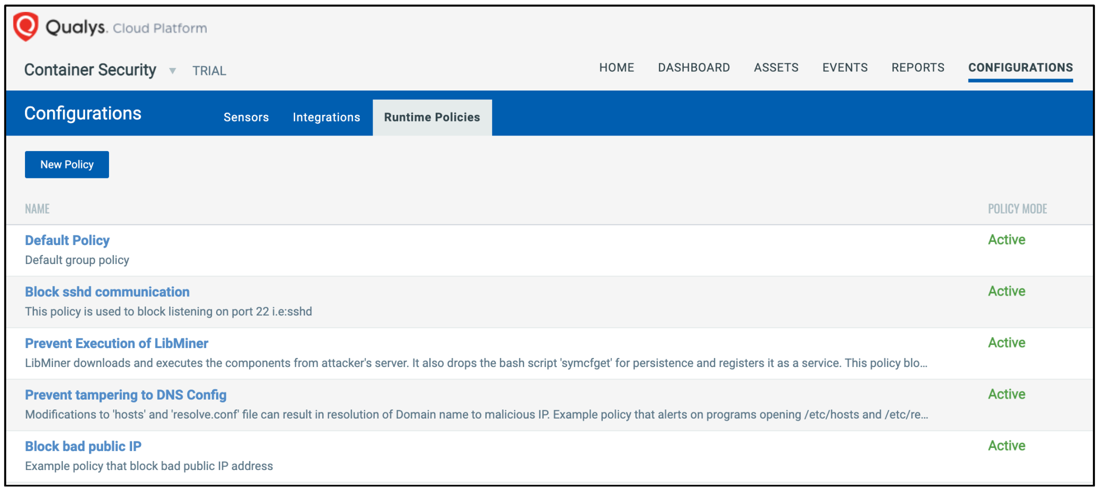
Navigate to the following URL to view the "Instrument Images from the Container Security UI" tutorial:

PLAY https://ior.ad/7Sng

Runtime Security Policy

A policy containing network, file, and\or application rule can be applied to an instrumented image.

Qualys provides sample policies to get you started. You can view these policies under the **Configurations** -> **Runtime Policies** tab.



NAME	POLICY MODE
Default Policy Default group policy	Active
Block sshd communication This policy is used to block listening on port 22 i.e:sshd	Active
Prevent Execution of LibMiner LibMiner downloads and executes the components from attacker's server. It also drops the bash script 'symcfget' for persistence and registers it as a service. This policy blo...	Active
Prevent tampering to DNS Config Modifications to 'hosts' and 'resolve.conf' file can result in resolution of Domain name to malicious IP. Example policy that alerts on programs opening /etc/hosts and /etc/re...	Active
Block bad public IP Example policy that block bad public IP address	Active

A runtime policy contains one or more rules of different types along with the mode the policy operates on, and the default action for each rule type.

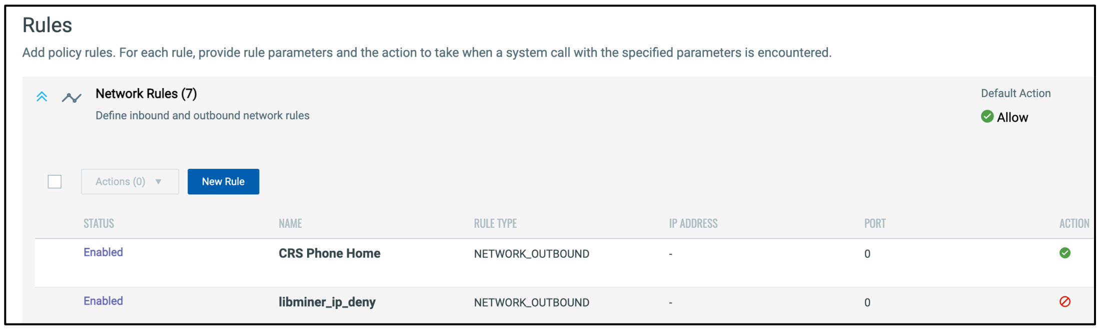
New policies can be created from scratch or by auto-generating a behavioral profile based policy from a running container.

Please note that the Behavioral Learning feature is available via the CRS API for advanced users.

Policy Rules

A policy rule can specify whether a program should or should not be able to execute a particular function (syscall), given a specific set of arguments.

For each rule, you need to provide rule parameters and the action to take when a system call with the specified parameters is encountered.



Rules

Add policy rules. For each rule, provide rule parameters and the action to take when a system call with the specified parameters is encountered.

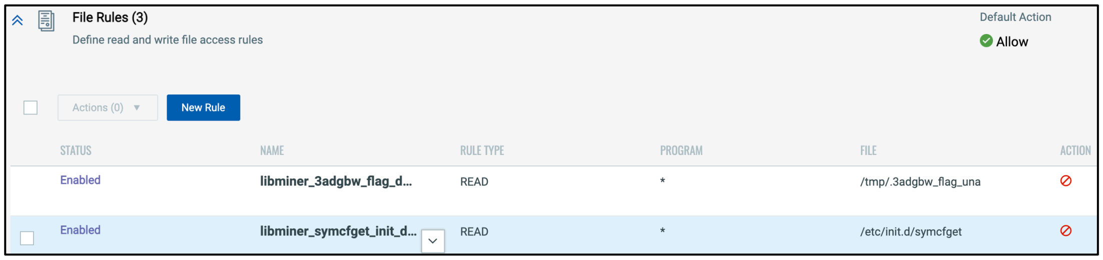
Network Rules (7)

Define inbound and outbound network rules

Default Action ✓ Allow

Actions (0) ▾ New Rule

STATUS	NAME	RULE TYPE	IP ADDRESS	PORT	ACTION
Enabled	CRS Phone Home	NETWORK_OUTBOUND	-	0	✓
Enabled	libminer_ip_deny	NETWORK_OUTBOUND	-	0	⊘



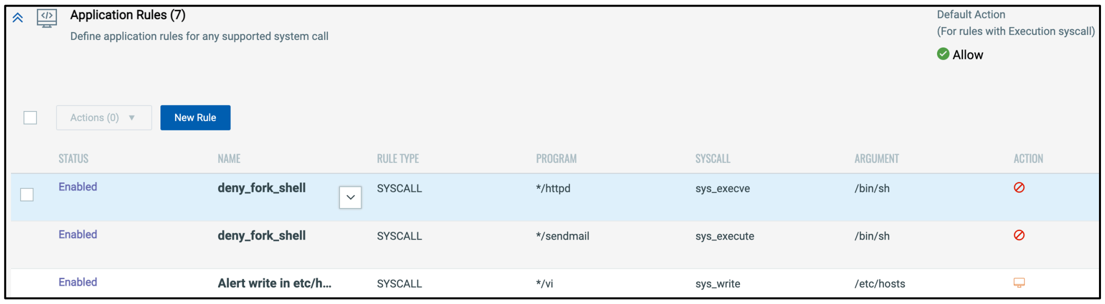
File Rules (3)

Define read and write file access rules

Default Action ✓ Allow

Actions (0) ▾ New Rule

STATUS	NAME	RULE TYPE	PROGRAM	FILE	ACTION
Enabled	libminer_3adgbw_flag_d...	READ	*	/tmp/.3adgbw_flag_una	⊘
Enabled	libminer_symcfget_init_d...	READ	*	/etc/init.d/symcfget	⊘



Application Rules (7)

Define application rules for any supported system call

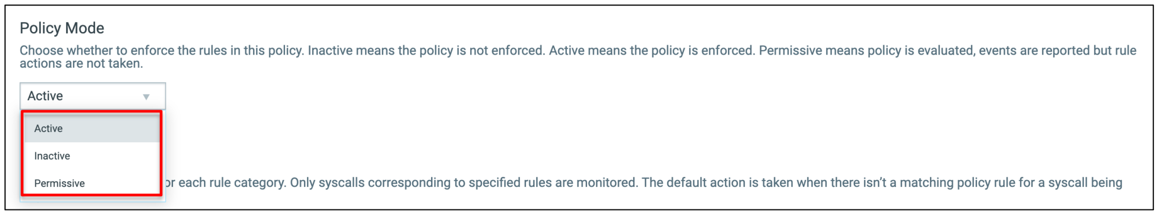
Default Action (For rules with Execution syscall) ✓ Allow

Actions (0) ▾ New Rule

STATUS	NAME	RULE TYPE	PROGRAM	SYSCALL	ARGUMENT	ACTION
Enabled	deny_fork_shell	SYSCALL	*/httpd	sys_execve	/bin/sh	⊘
Enabled	deny_fork_shell	SYSCALL	*/sendmail	sys_execute	/bin/sh	⊘
Enabled	Alert write in etc/h...	SYSCALL	*/vi	sys_write	/etc/hosts	🖵

You can choose default actions for Network, File and Application rule types. This is the default action that will be taken unless there is a policy rule that overwrites this action.

You can also define a list of system calls that you want to ignore for the policy. When a system call is ignored, no new events will be created for the system call even if it matches one of the policy rules.



Policy Mode

Choose whether to enforce the rules in this policy. Inactive means the policy is not enforced. Active means the policy is enforced. Permissive means policy is evaluated, events are reported but rule actions are not taken.

Active ▾

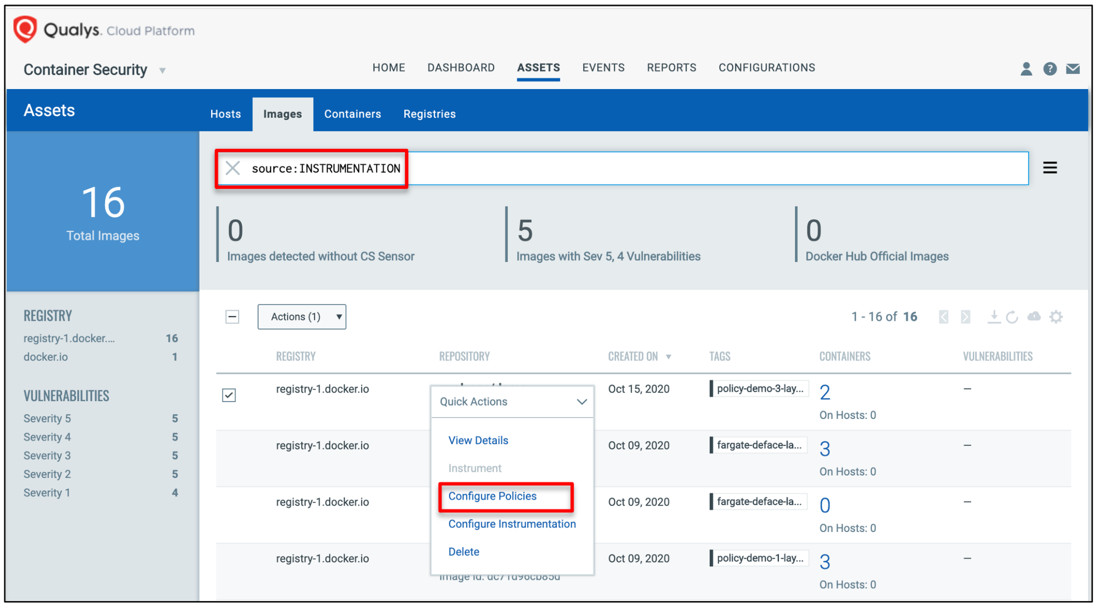
or each rule category. Only syscalls corresponding to specified rules are monitored. The default action is taken when there isn't a matching policy rule for a syscall being

You can choose a policy enforcement mode (Active, Inactive, Permissive).
The option you pick determines whether or not the policy rules will be enforced on the containers that are spawned from the image. The policy is enforced only when **Active** is

selected. When **Permissive** is selected, the events are reported but actions are not enforced. Note that you can change this at any time after the policy is saved.

Apply Policy

You can apply a policy to an instrumented image to enforce certain behavioral restrictions to secure the container spawned from the image.



source:INSTRUMENTATION

16 Total Images

0 Images detected without CS Sensor

5 Images with Sev 5, 4 Vulnerabilities

0 Docker Hub Official Images

REGISTRY	
registry-1.docker....	16
docker.io	1

VULNERABILITIES	
Severity 5	5
Severity 4	5
Severity 3	5
Severity 2	5
Severity 1	4

1 - 16 of 16

REGISTRY	REPOSITORY	CREATED ON	TAGS	CONTAINERS	VULNERABILITIES
registry-1.docker.io		Oct 15, 2020	policy-demo-3-lay...	2 On Hosts: 0	–
registry-1.docker.io		Oct 09, 2020	fargate-deface-la...	3 On Hosts: 0	–
registry-1.docker.io		Oct 09, 2020	fargate-deface-la...	0 On Hosts: 0	–
registry-1.docker.io		Oct 09, 2020	policy-demo-1-lay...	3 On Hosts: 0	–

When a container is spun up from such an instrumented image, it inherits the policy settings from the instrumented image.

To find an instrumented container, go to **Assets** -> **Containers** in the Container Security application and perform a search using this search query:
`isInstrumented: true`

We can then monitor and detect syscalls when they hit the glibc, and activities that are in violation of the policy can be identified. If the policy is active, we can stop malicious activity in the container before the function completes its task(s).

Configure Instrumentation (LogMode)

Once a policy is applied to an image you can choose a LogMode to determine what is logged in a container for policy hits (rule matches) and behaviour.

You can choose from the following LogMode options to determine which policy hit events should be logged:

- None - No events get logged.
- PolicyMonitor - Only policy hit events with Monitor action.
- PolicyDeny - Only policy hit events with Deny action.
- PolicyMonitorDeny - Only policy hit events with Monitor or Deny action.
- PolicyAllow - Only policy hit events with Allow action.
- PolicyAll - Only policy hit events with Monitor, Deny or Allow action.
- Behavior - Only detailed behavioral events.
- All - Includes events that match PolicyAll plus events that match Behavior.

You can apply a policy and configure Instrumentation (Log mode) only to images when using the CS UI. This way when you apply a configuration to the container image, all the containers spawned from that image are secured and adhere to your configuration. A change to the configuration assigned to the image will be applied to all the running containers.

CRS API

When using the CRS API, you can also apply a policy and LogMode configuration directly to container in the absence of an image assigned policy. More parameters are available via the API.

Navigate to the following URL to view the "Runtime Security Policy" tutorial:

PLAY https://ior.ad/7SuF

Verify CRS Functionality

After instrumenting an image and applying a policy, you can spin up a container from the protected image to verify CRS functionality.

If the instrumented image is in a registry, you will first need to pull the image to a Docker host using the following command:

```
$ docker pull myregistry/centos:latest-layered
```

where `myregistry/centos:latest-layered` is the instrumented or protected image.

Instantiate Container

You can spin up a container from the protected image using the following command:

```
$ docker run -itd -e
LIMQURL=https://gateway.qg3.apps.qualys.com/crs/v1.2 -e
LI_MQSKIPVERIFYTLS=true cstraining/centos:centos7-layered
```

where https://gateway.qg3.apps.qualys.com is your Qualys gateway URL, which depends on your subscription.

When you spawn a container from the instrumented image, the policy applied to the instrumented image gets enforced on the container.

You can then connect to the container using the following command:.

```
$ docker exec -it f5da5c286ddb /bin/bash
```

where `f5da5c286ddb` is the “Container ID” of the protected container.

When a container is spun up from the protected image, you can test various scenarios within the container and verify the protection works as desired in your environment.

You can then view information on policy hits, syscalls and action details in the container Runtime Profile in the Container Security application. And the behaviour log (if enabled on the image) will reflect a policy violation.

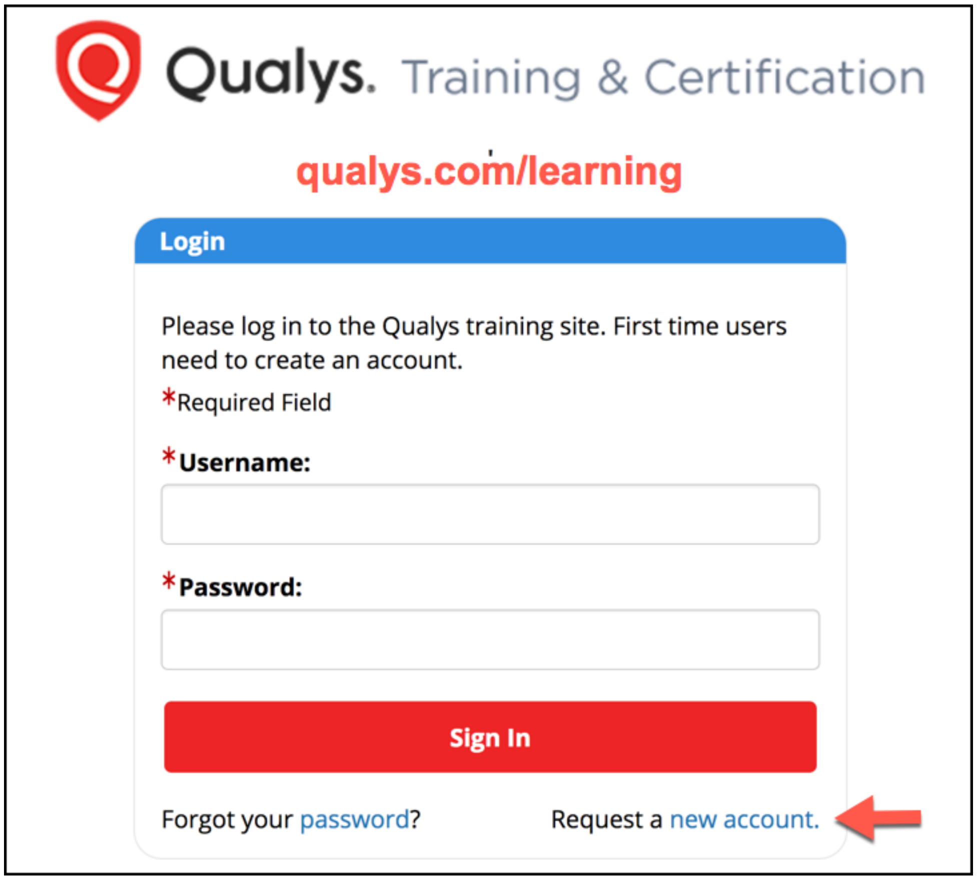
Navigate to the following URL to view the “Verify CRS Functionality” tutorial:

PLAY https://ior.ad/7Su8

Container Security Certification Exam

Participants in this training course have the option to take the Container Security Certification Exam. This exam is provided through our Learning Management System (qualys.com/learning).

To take the exam, candidates will need a "learner" account.



Please log in to the Qualys training site. First time users need to create an account.

*Required Field

*Username:

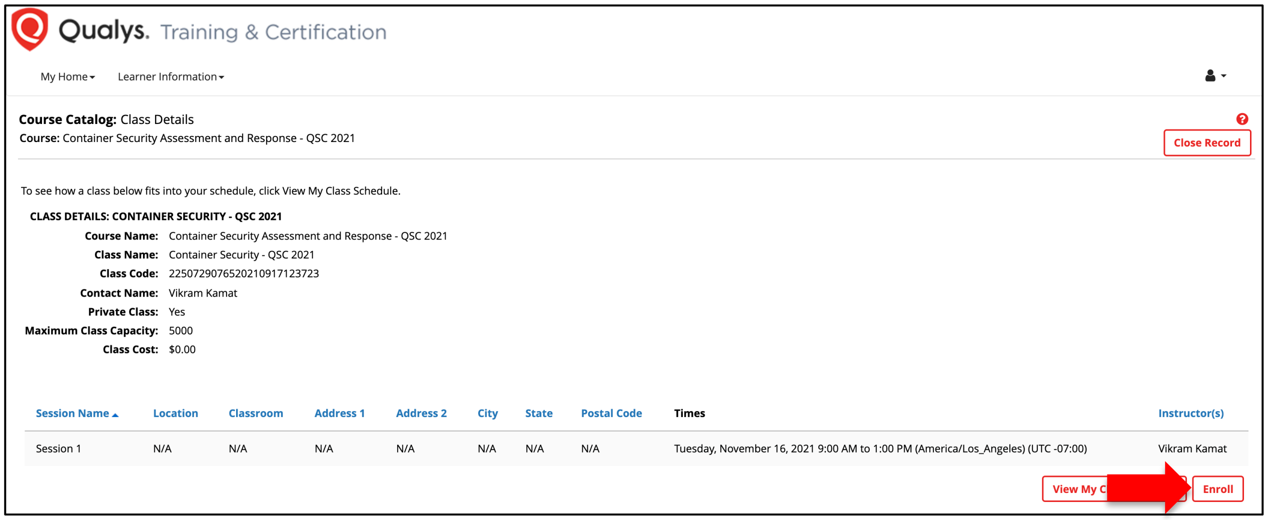
*Password:

Forgot your password? Request a new account.

If you would like to take the exam, but do not already have a "learner" account, click the "Request a new account" link, from the "Qualys Training & Certification" login page (qualys.com/learning).

Once you have created a "learner" account (and for those who already have an account), click the following link to access the "Container Security - QSC 2021" course page:

https://gm1.geolearning.com/geonext/qualys/scheduledclassdetails4enroll.geo?&id=22511237814



Course Catalog: Class Details
Course: Container Security Assessment and Response - QSC 2021

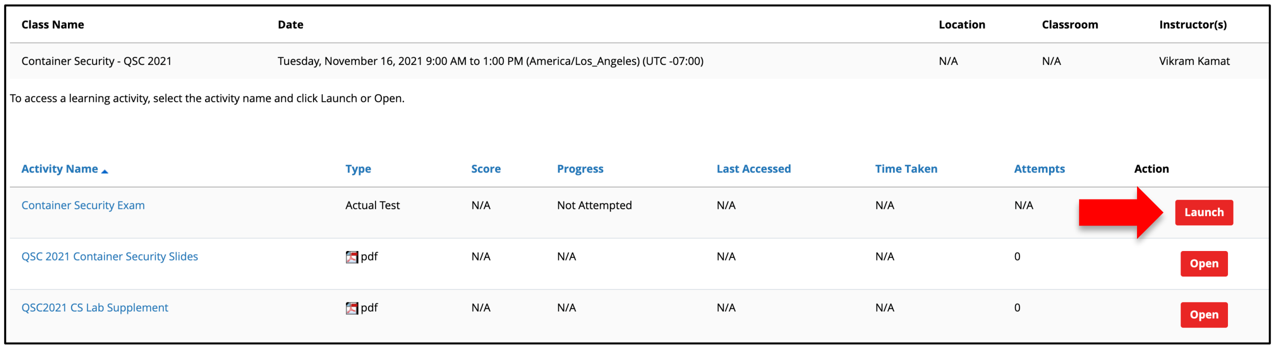
To see how a class below fits into your schedule, click View My Class Schedule.

CLASS DETAILS: CONTAINER SECURITY - QSC 2021

Course Name:	Container Security Assessment and Response - QSC 2021
Class Name:	Container Security - QSC 2021
Class Code:	2250729076520210917123723
Contact Name:	Vikram Kamat
Private Class:	Yes
Maximum Class Capacity:	5000
Class Cost:	$0.00

Session Name	Location	Classroom	Address 1	Address 2	City	State	Postal Code	Times	Instructor(s)
Session 1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Tuesday, November 16, 2021 9:00 AM to 1:00 PM (America/Los_Angeles) (UTC -07:00)	Vikram Kamat

From the "Container Security - QSC 2021" course page, click the "Enroll" button (lower-right corner).



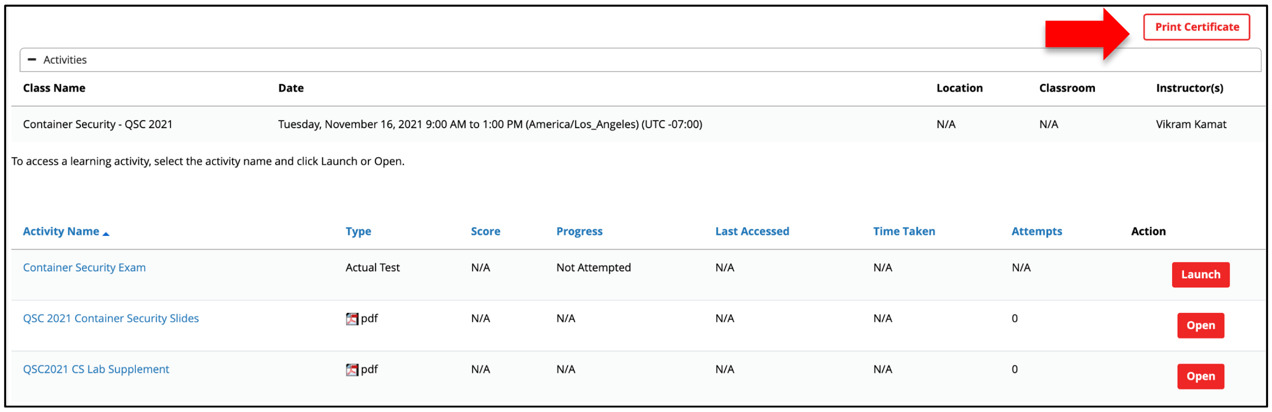
Class Name	Date	Location	Classroom	Instructor(s)
Container Security - QSC 2021	Tuesday, November 16, 2021 9:00 AM to 1:00 PM (America/Los_Angeles) (UTC -07:00)	N/A	N/A	Vikram Kamat

To access a learning activity, select the activity name and click Launch or Open.

Activity Name	Type	Score	Progress	Last Accessed	Time Taken	Attempts	Action
Container Security Exam	Actual Test	N/A	Not Attempted	N/A	N/A	N/A	Launch
QSC 2021 Container Security Slides	pdf	N/A	N/A	N/A	N/A	0	Open
QSC2021 CS Lab Supplement	pdf	N/A	N/A	N/A	N/A	0	Open

After completing the course enrollment, click the "Launch" button, for the Qualys Container Security Certification Exam.

Each candidate is provided five attempts to pass the exam.



Class Name	Date	Location	Classroom	Instructor(s)
Container Security - QSC 2021	Tuesday, November 16, 2021 9:00 AM to 1:00 PM (America/Los_Angeles) (UTC -07:00)	N/A	N/A	Vikram Kamat

To access a learning activity, select the activity name and click Launch or Open.

Activity Name	Type	Score	Progress	Last Accessed	Time Taken	Attempts	Action
Container Security Exam	Actual Test	N/A	Not Attempted	N/A	N/A	N/A	Launch
QSC 2021 Container Security Slides	pdf	N/A	N/A	N/A	N/A	0	Open
QSC2021 CS Lab Supplement	pdf	N/A	N/A	N/A	N/A	0	Open

With a passing score of 75% (or greater), click the “Print Certificate” button to download and print your course exam certificate.